Dialogic®

Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 4.1

Installation and Configuration Guide
with TeleStax Apache-Tomcat Application Server

May 2015 Rev 1.2

www.dialogic.com

Copyright and Legal Notice

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see http://www.dialogic.com/company/terms-of-use.aspx for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

Any use case(s) shown and/or described herein represent one or more examples of the various ways, scenarios or environments in which Dialogic® products can be used. Such use case(s) are non-limiting and do not represent recommendations of Dialogic as to whether or how to use Dialogic products.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Table of Contents

Revision History

Revision	Release Date	Notes
1.2 (Updated)	May 2015	Installation and Configuration: • Added Configuration of WAR File via AS Web Admin Console and Placement of WAR File on AS to the Deploy JSR 309 Connector Demo Application section.
1.2	April 2015	Updates to support JSR 309 Connector Release 4.1. Updates to support log4j2 implementation. JSR 309 Connector Requirements: • Updated the supported version to PowerMedia XMS Release 2.4 Service Update 1. Contents of the Distribution: • Updated the Distributed Files section. Appendix C: Updating the JSR 309 Connector: • Updated with details on the *MANIFEST.MF* file.
1.1	February 2015	Appendix B: Redundant Media Servers Configuration: • Updated with details on hot active/standby redundancy. Appendix C: Updating the JSR 309 Connector: • Added new section.
1.0	October 2014	Initial version of document.
Last modified: May 2015		

1. JSR 309 Connector Requirements

The following requirements are needed to be in place before installing the JSR 309 Connector:

- A functional TeleStax Apache-Tomcat Application Server platform for development and testing.

  The JSR 309 Connector has been tested with the following Apache-Tomcat versions of TeleStax Application Server.

  - TeleStax Mobicents Apache-Tomcat AS:

    mss-3.0.536-apache-tomcat-7.0.50

  - TeleStax TelScale Apache-Tomcat AS:

    TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50

  Note: Refer to www.telestax.com for additional information about TeleStax Application Server and their licensing.

- A functional PowerMedia XMS Release 2.4 Service Update 1 system.

  Note: Refer to Proper Configuration of PowerMedia XMS for additional information.

- SIP phones or soft clients.

2. Contents of the Distribution

This section lists and describes the files in the JSR 309 Connector distribution.

Distributed Files

The JSR 309 Connector distribution consists of a single TAR file:

TeleStaxTomcat-msc#.#.tar

This package contains the following structure:

JSR 309 Connector Files	Description
DIR: */DlgcJSR309/application/* CONTENTS: *deploymentDescriptor/* *sample-src/* *build.xml* *dlgmsc_tests.war*	Directory that contains a deployable web archive that can be used to test the supported functionality. The WAR file implements several test servlets. Refer to Test Servlets for more information. It also contains the test servlets source files and build environment in order to simply create demo application project.
DIR: */DlgcJSR309/lib/* CONTENTS: *dlgmsc.jar* *msmltypes.jar* *dlgsmiltypes.jar* *<third-party required files>*	Directory that contains the JSR 309 Connector implementation for PowerMedia XMS which consists of a sets of 3 JAR files: *dlgmsc.jar*, *msmltypes.jar*, and *dlgcsmiltypes.jar*. Directory also contains additional third-party libraries required by JSR 309 Connector.
DIR: */DlgcJSR309/properties/* CONTENTS: *dlgc_JSR309.properties* *dlgc_demos.properties* *log4j2.xml*	Directory that contains the properties files used to set up configuration for JSR 309 Connector and provided demos as well as xml configuration file for logging framework.
DIR: */DlgcJSR309DemoPrompts/* CONTENTS: *prompts.tar*	JSR 309 Connector prompts used by demo application. Refer to Installing the Demo Prompts for further details.

3. Installation and Configuration

This section describes how to install and use the JSR 309 Connector.

For system requirements and supported platforms, see JSR 309 Connector Requirements.

Preparing the J2EE Converged Application Server

The JSR 309 Connector has been deployed and tested on specific versions of TeleStax Apache-Tomcat Application Servers. For quick instructions on how to install and configure desired AS for JSR 309 Connector usage, refer to Appendix A: JSR 309 Connector Environment Setup.

Installing the JSR 309 Connector

The JSR 309 Connector is a library used by an application which needs to be configured within Application Server itself.

The JSR 309 Connector demo applications provided with the distribution are used to illustrate some functionality of the JSR 309 Connector. Refer to Test Servlets for further details.

The following steps are necessary for JSR 309 Connector and demo application installation for correct operation:

- Step 1 – Installation and Configuration of JSR 309 Connector
- Step 2 – Installation and Configuration of JSR 309 Connector Demo
- Step 3 – Proper Configuration of PowerMedia XMS
- Step 4 – Verification of JSR 309 Connector using Demo Application

You need to extract the distribution package as various components (files) will be needed to correctly complete each step. Refer to Contents of the Distribution which describes the contents in detail.

Step 1 – Installation and Configuration of JSR 309 Connector

Simply place the package TAR file on TeleStax Apache-Tomcat Linux server and run the following command:

```
tar -xvf TeleStaxTomcat-msc#.#.tar
```

This will create two directories, "*DlgcJSR309*" and "*DlgcJSR309DemoPrompts*", as described in Contents of the Distribution.

Note: These directories are referenced throughout this document for content required by JSR 309 Connector.

Follow these steps to properly configure JSR 309 Connector:

- Modify AS startup script – *catalina.sh*.
- Configure JSR 309 Connector properties file.

Modify AS Startup Script

Edit the following examples and then add the section marked below in **bold**:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/bin/catalina.sh

(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/bin/catalina.sh
```

```
# Uncomment the following line to make the umask available when using the
# org.apache.catalina.security.SecurityListener
#JAVA_OPTS="$JAVA_OPTS -Dorg.apache.catalina.security.SecurityListener.UMASK=`umask`"
# ----- Execute The Requested Command -----------------------------------------

#Dialogic Additions
export APPSERVER_PLATFORM="TELESTAX"
export DLG_PROPERTY_FILE=${CATALINA_HOME}/conf/Dialogic/dlgc_JSR309.properties

# Bugzilla 37848: only output this if we have a TTY
if [ $have_tty -eq 1 ]; then
echo "Using CATALINA_BASE:   $CATALINA_BASE"
```

Note: For Remote Debugging, the firewall needs to be configured to allow port access for Remote Socket Debugging to work.

Configure JSR 309 Connector Properties File

The *dlgc_JSR309.properties* file is used to configure the location (IP addresses and ports) of the Application Server using JSR 309 Connector and the PowerMedia XMS Media Server to be used by connector. Follow these steps to configure the logging facility in the Application Server platform:

1. From the extracted distribution package, copy the *dlgc_JSR309.properties* file from the "*DlgcJSR309/properties*" directory to the Application Server under a new directory:

   ```
   (TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/conf/Dialogic

   (Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/conf/Dialogic
   ```

2. Edit the *dlgc_JSR309.properties* file according to your Application Server and PowerMedia XMS configuration.
   - The changes will include the AS IP address and port of the SipServlet container running the JSR 309 Connector.
   - Changes will also include the PowerMedia XMS IP address and port.

```
…
# Connector's address information (Typically same as the SipServlet container) your AS IP Address
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5080

….
#Media Server
mediaserver.msType=XMS
mediaserver.1.sip.address=xxx.xxx.xxx.xxx
mediaserver.1.sip.port=5060

….
```

The JSR 309 Connector is now configured.

Step 2 – Installation and Configuration of JSR 309 Connector Demo

At this point, an application can take advantage of JSR 309 Connector and use its resources for media related functionality. The JSR 309 Connector package provides a demo application which uses JSR 309 Connector to illustrate various media functionalities. This step will illustrate how to install and configure JSR 309 Connector demo application. Step 3 will illustrate how to verify that the demo application works with JSR 309 Connector and communicates correctly with PowerMedia XMS.

Follow these steps to set JSR 309 Connector demo application:

1. Modify AS startup script – *catalina.sh*.
2. Configure JSR 309 Connector Demo properties file.
3. Deploy JSR 309 Connector Demo application.

Modify AS Startup Script

Edit the following examples:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/bin/catalina.sh

(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/bin/catalina.sh
```

Then, file and add the section marked in **bold**:

```
# Uncomment the following line to make the umask available when using the
# org.apache.catalina.security.SecurityListener
#JAVA_OPTS="$JAVA_OPTS -Dorg.apache.catalina.security.SecurityListener.UMASK=`umask`"
# ----- Execute The Requested Command -----------------------------------------

#Dialogic Additions
export APPSERVER_PLATFORM="TELESTAX"
export DIALOGIC_DEMO_PROPERTY_FILE=${CATALINA_HOME}/conf/Dialogic/dlgc_demos.properties
export DLG_PROPERTY_FILE=${CATALINA_HOME}/conf/Dialogic/dlgc_JSR309.properties

# Bugzilla 37848: only output this if we have a TTY
if [ $have_tty -eq 1 ]; then
  echo "Using CATALINA_BASE:   $CATALINA_BASE"
```

Configure JSR 309 Connector Demo Properties File

The demo properties file has various settings for various sample applications that can be modified. For detailed information on various configurations, refer to the descriptions of each sample application in Test Servlets.

From the distribution package under "*DlgcJSR309/properties*", copy the *dlgc_demos.properties* file to the following directory:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/conf/Dialogic

(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/conf/Dialogic
```

Deploy JSR 309 Connector Demo Application

Next, the JSR 309 Connector demo application needs to be deployed in the TeleStax Application Server. To do so, proceed to the following instructions.

There are two ways to deploy an application WAR file in TeleStax Application Server. One is by using AS web management console and the other is by placing WAR file directly on the server.

Configuration of WAR File via AS Web Admin Console

Start TeleStax Application Server by running startup script:

- From:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/bin/
(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/bin/
```

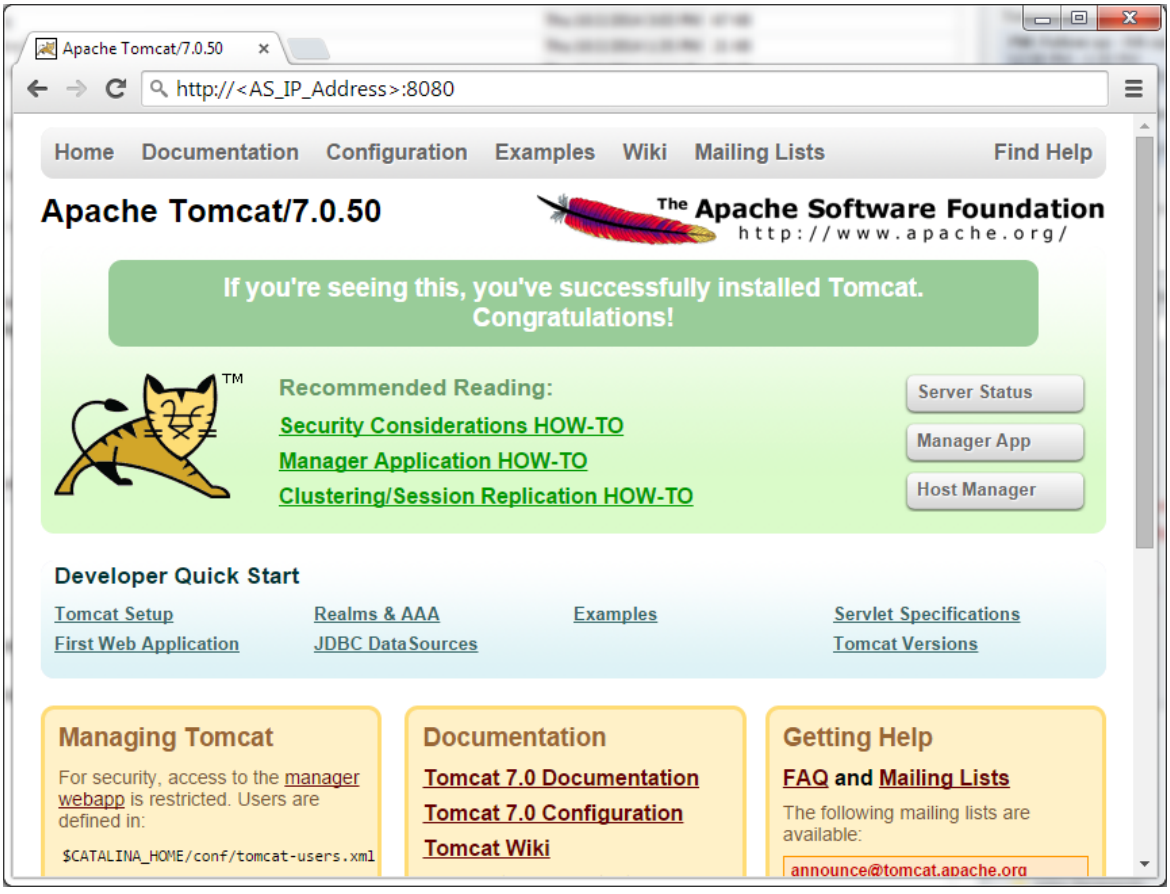
- Run:

```
./catalina.sh run
```

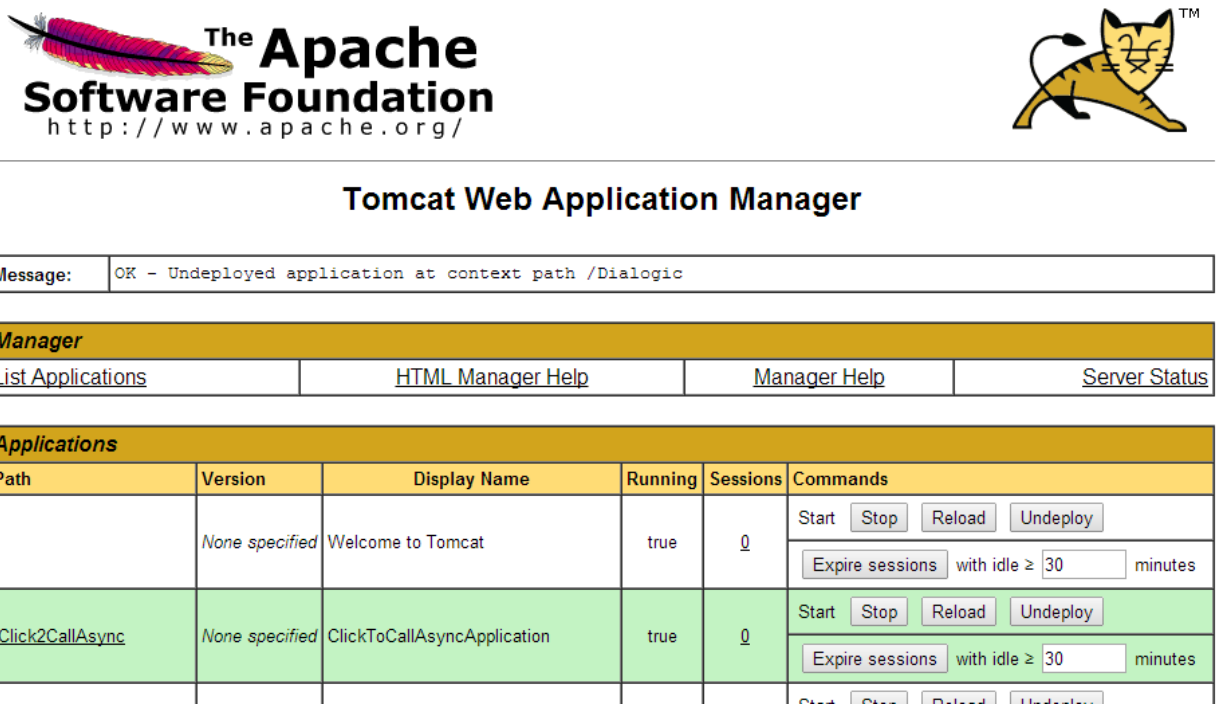
```
agement.war"
12:51:59,059 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "click2call.war"
12:51:59,059 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "jolokia.war"
12:51:59,059 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "media-jsr309-ser
vlet.war"
12:51:59,060 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 29) JBAS018559: Deployed "websockets-sip-s
ervlet.war"
12:51:59,082 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015961: Http management interface listening on htt
p://146.152.122.177:9990/management
12:51:59,082 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on http://146.152.
122.177:9990
12:51:59,083 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS 7.1.3.Final "Arges" started in 42
00ms - Started 377 of 474 services (96 services are passive or on-demand)
```

Provided that the server has started without any errors/exceptions, the application is ready to be deployed.

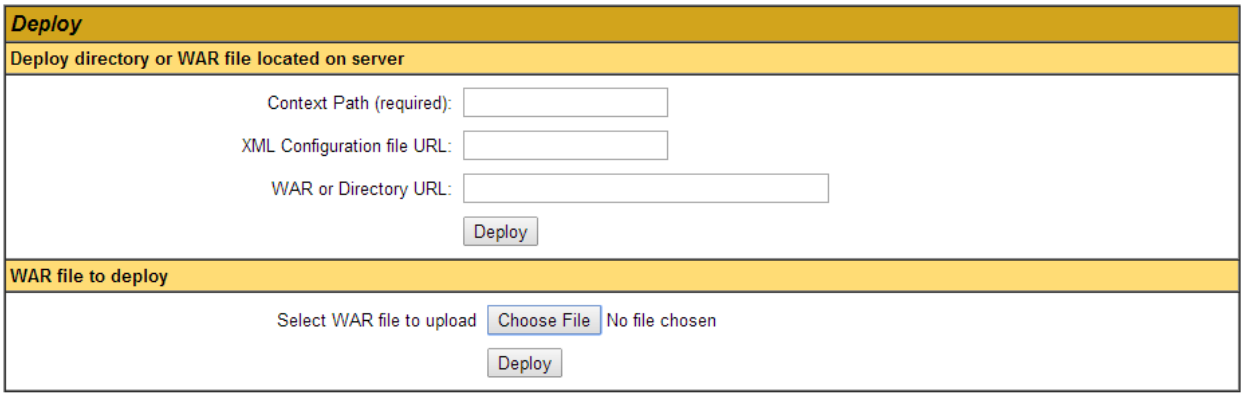
Go to http://<as_ip_address>:8080 then provide appropriate credentials as defined during the installation process.



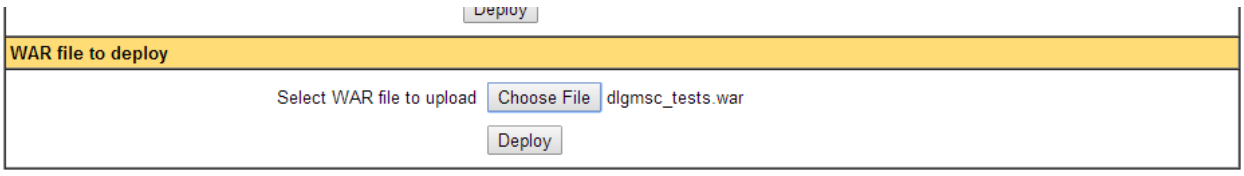
Click on **Manager App** button on the top right of your screen.



Scroll down to the **Deploy** section:

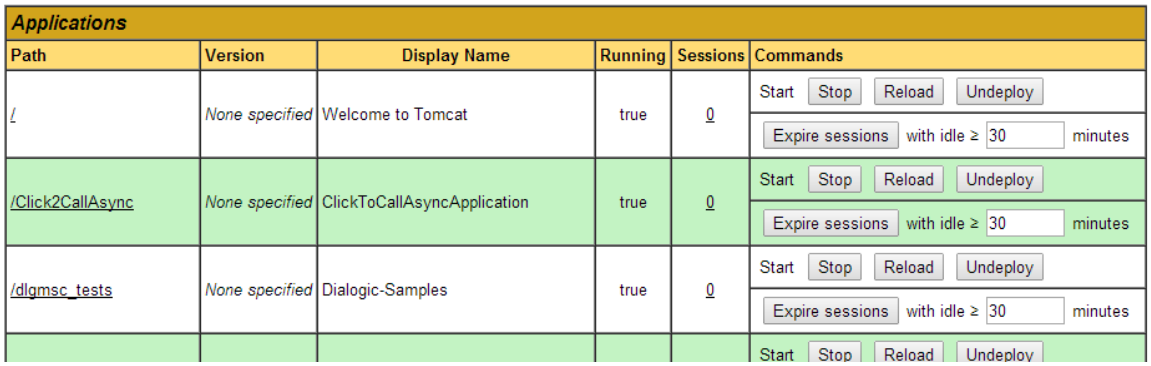


Under the **WAR file to deploy** section, click on **Choose File** and select *dlgmsc_tests.war* file as provided in distribution under "*TeleStaxTomcat-msc#.#/DlgcJSR309/applications*" directory.



Click on **Deploy** button.

Once successfully deployed, the application "*/dlgmsc_tests*" will appear under the **Applications** section at the top of the page.



Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/Click2CallAsync	None specified	ClickToCallAsyncApplication	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/dlgmsc_tests	None specified	Dialogic-Samples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
					Start Stop Reload Undeploy

The **Console** window will also indicate the successful deployment.

```
2014-04-02 20:47:04,826 WARN  [DlgcXMsControlFactory] (http-bio-8080-exec-11) Assuming TCK is not enabled
2014-04-02 20:47:04,933 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11)
org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl@4eed6301 the following sip servlet application has been added : Dialogic-
Samples
2014-04-02 20:47:04,933 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) It contains the following Sip Servlets :
2014-04-02 20:47:04,933 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcPlayerPerformanceTest
2014-04-02 20:47:04,933 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcRecorderTest
2014-04-02 20:47:04,933 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcMsMonitorServlet
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcReferenceConferenceWithOutBCallServlet
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcDtmfPromptCollectTest
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : JMCConferenceServlet
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcBridgeServlet
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcDtmfAsyncTest
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
Samples/ServletName : DlgcEMBridgeServlet
2014-04-02 20:47:04,934 INFO  [SipApplicationDispatcherImpl] (http-bio-8080-exec-11) SipApplicationName : Dialogic-
```

Placement of WAR File on AS

Another way to deploy the application is to place the WAR file directly in the following directory, which will automatically be recognized and started:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/webapps

(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/webapps
```

Step 3 – Proper Configuration of PowerMedia XMS

In order to verify the correct JSR 309 Connector installation with provided JSR 309 Connector demos, you will need to correctly configure PowerMedia XMS Media Server. This includes:

- PowerMedia XMS Web Admin Configuration.
  - Allowing Absolute Paths
  - Installing JSR 309 Connector Demo Prompts
- Demo required prompts installed on PowerMedia XMS itself (optional).

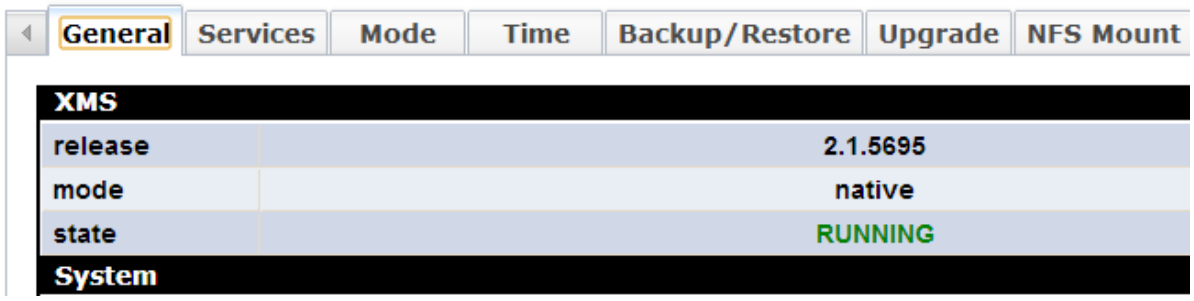
  Note: Only needed if JSR 309 Connector demo application is going to be used as it depends on these prompts to be installed on PowerMedia XMS.

PowerMedia XMS Web Admin Configuration

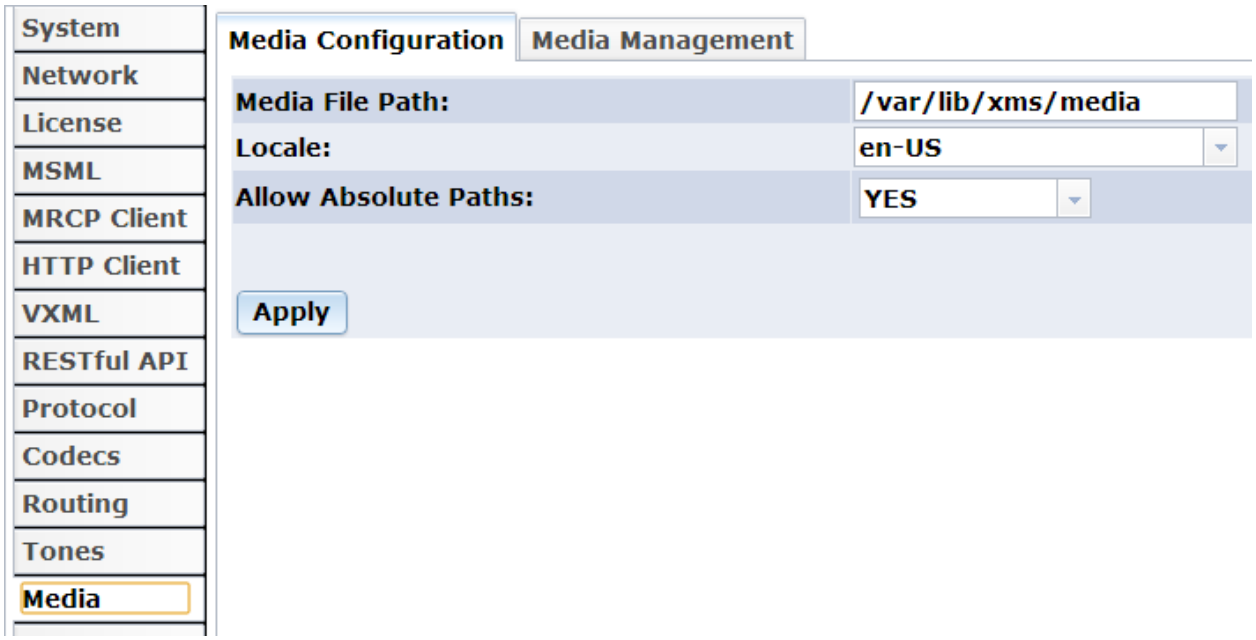
Allowing Absolute Paths

JSR 309 Connector uses Native MSML interface to PowerMedia XMS Media Server. You need to verify that PowerMedia XMS is indeed configured for "Native" mode.

Note: In PowerMedia XMS Release 2.1 and later, "Native" mode is the mode configured by default when PowerMedia XMS gets installed. Also, it is strongly recommended that the latest version of PowerMedia XMS be used.



Now, under the **Media** menu, click on **Media Configuration** tab. The **Allow Absolute Paths** field must be set to YES.

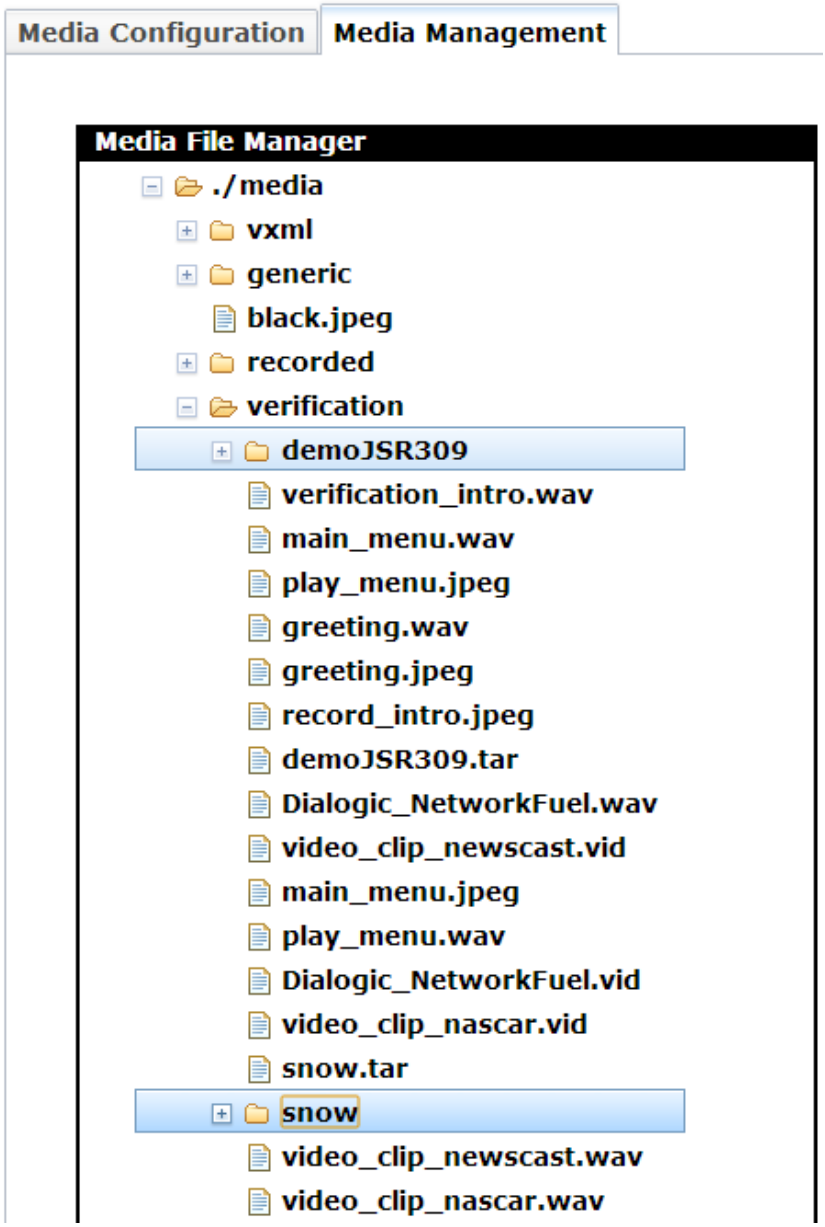


Once appropriate changes are made, click the **Apply** button which will commit the changes. Once changes are applied, you will be asked to restart PowerMedia XMS. This step is not yet required since we are going to be changing more configuration parameters below.

Installing JSR 309 Connector Demo Prompts

Custom prompts need to be installed on PowerMedia XMS as the various Dialogic demos will require them to work.

Once installed, they should appear in the **Media** menu under the **Media Management** tab as shown below in the highlighted fields:



You can locate and install the demo prompts by performing the following:

1. Copy the *prompts.tar* file inside the "*<Release Package>/DlgcJSR309DemoPrompts*" directory to the PowerMedia XMS system under the "*/var/lib/xms/media/en_US/verification*" directory.
2. Untar file using the command:

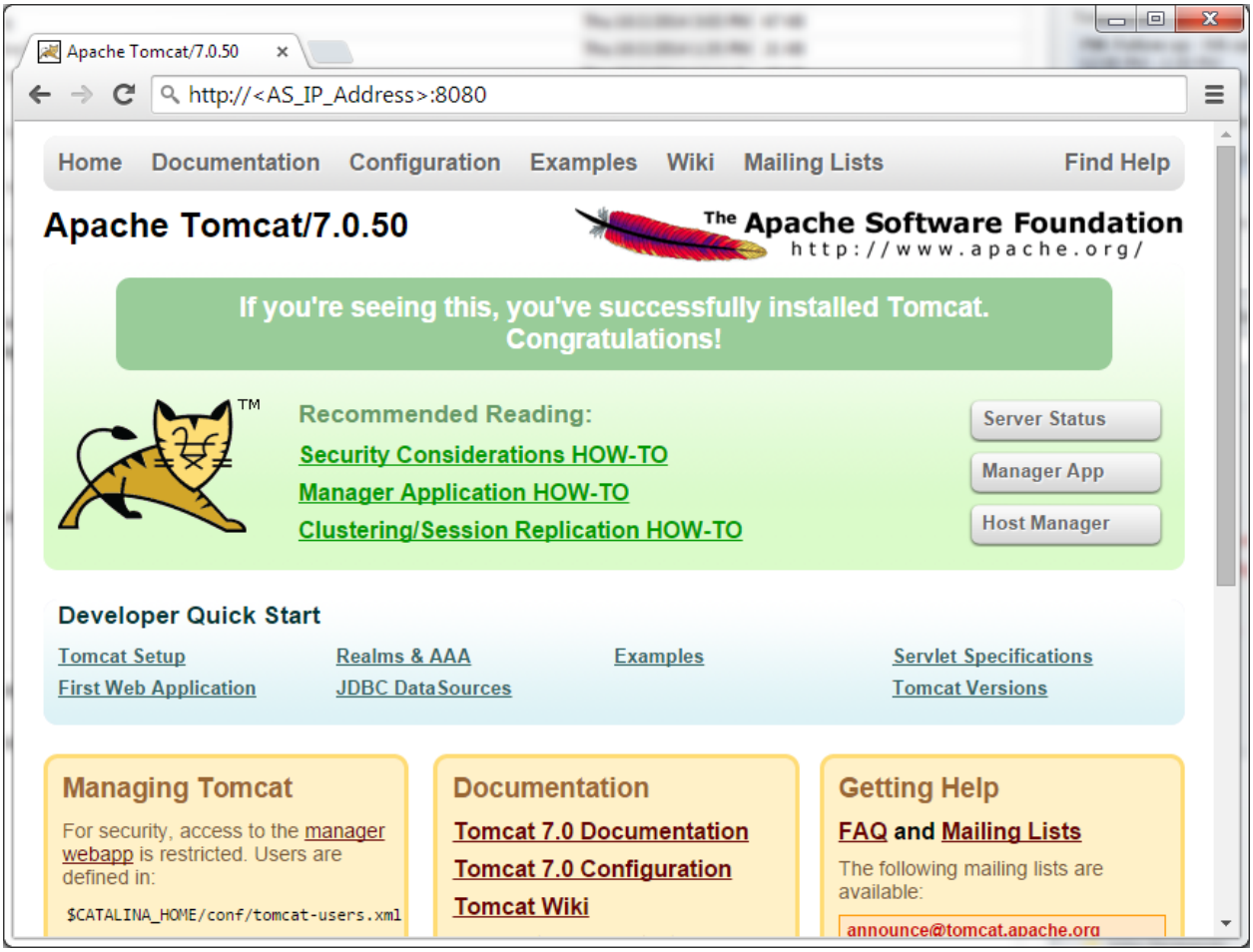
```
tar -xvf <file_name>.tar
```

Step 4 – Verification of JSR 309 Connector using Demo Application

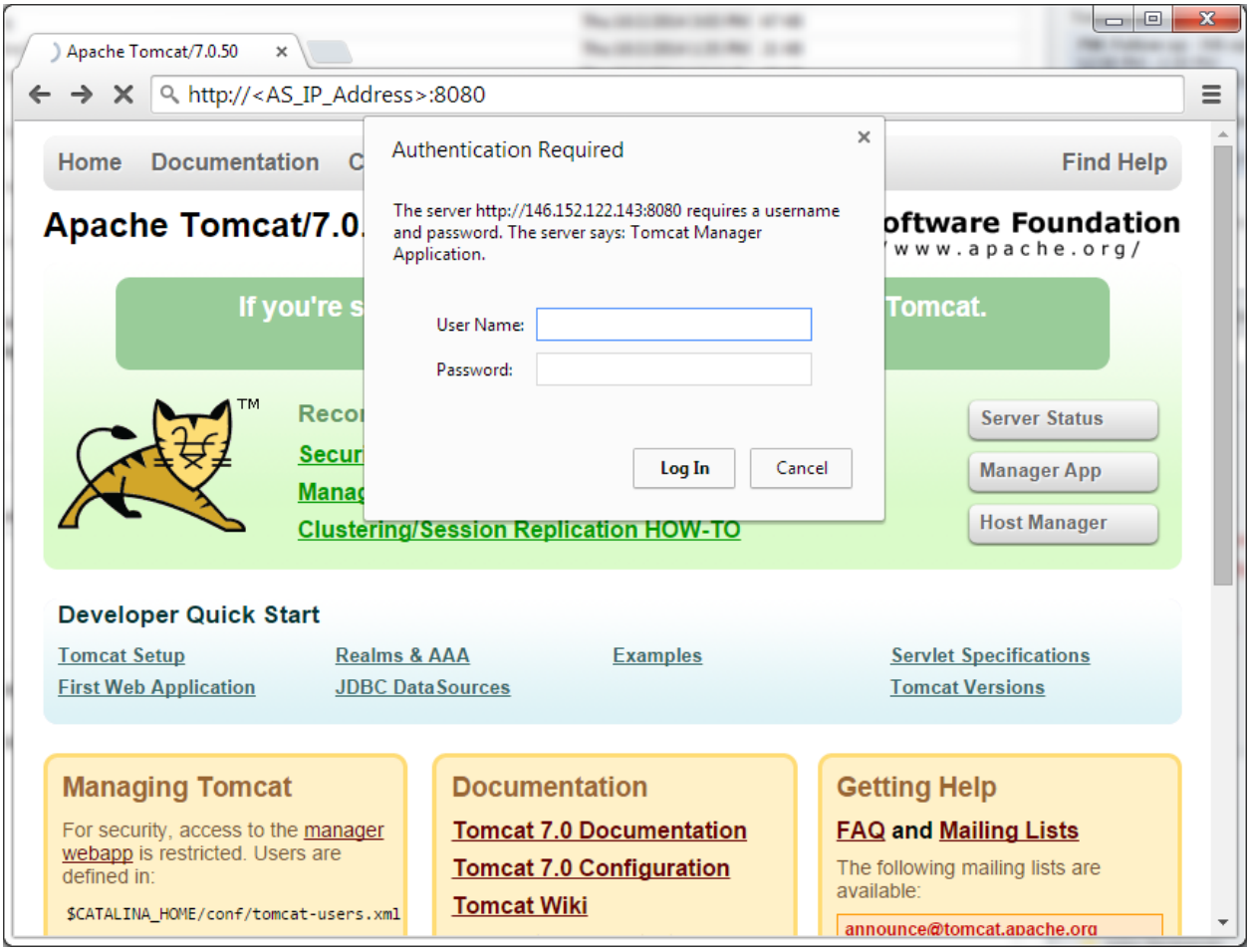
With *default dlgc_demos.properties* file, you can use a simple ***DlgcPlayerDemo***, which is part of supplied demo application WAR file. This demo simply answers an incoming call and uses JSR 309 Connector to request media resources from PowerMedia XMS in order to play an audio file.

Before the deployed application can process SIP messages, it needs to be configured in SIP Servlets Management console:

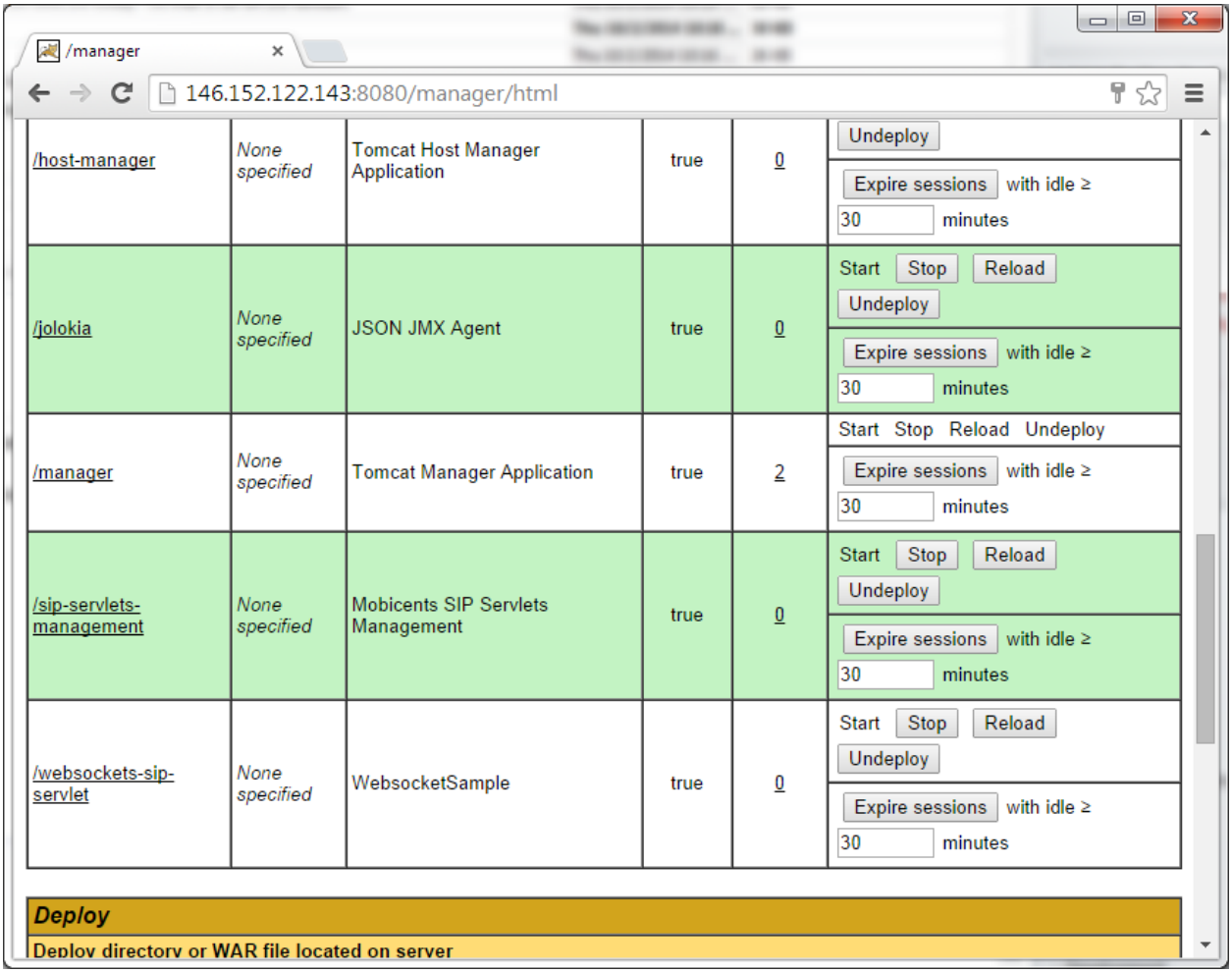
- Go to http://<as_ip_address>:8080.
- Log in with set up credentials as defined in tomcat-users.xml as described in Appendix A: JSR 309 Connector Environment Setup under TeleStax Verification (for example, username and password was set to admin/admin).



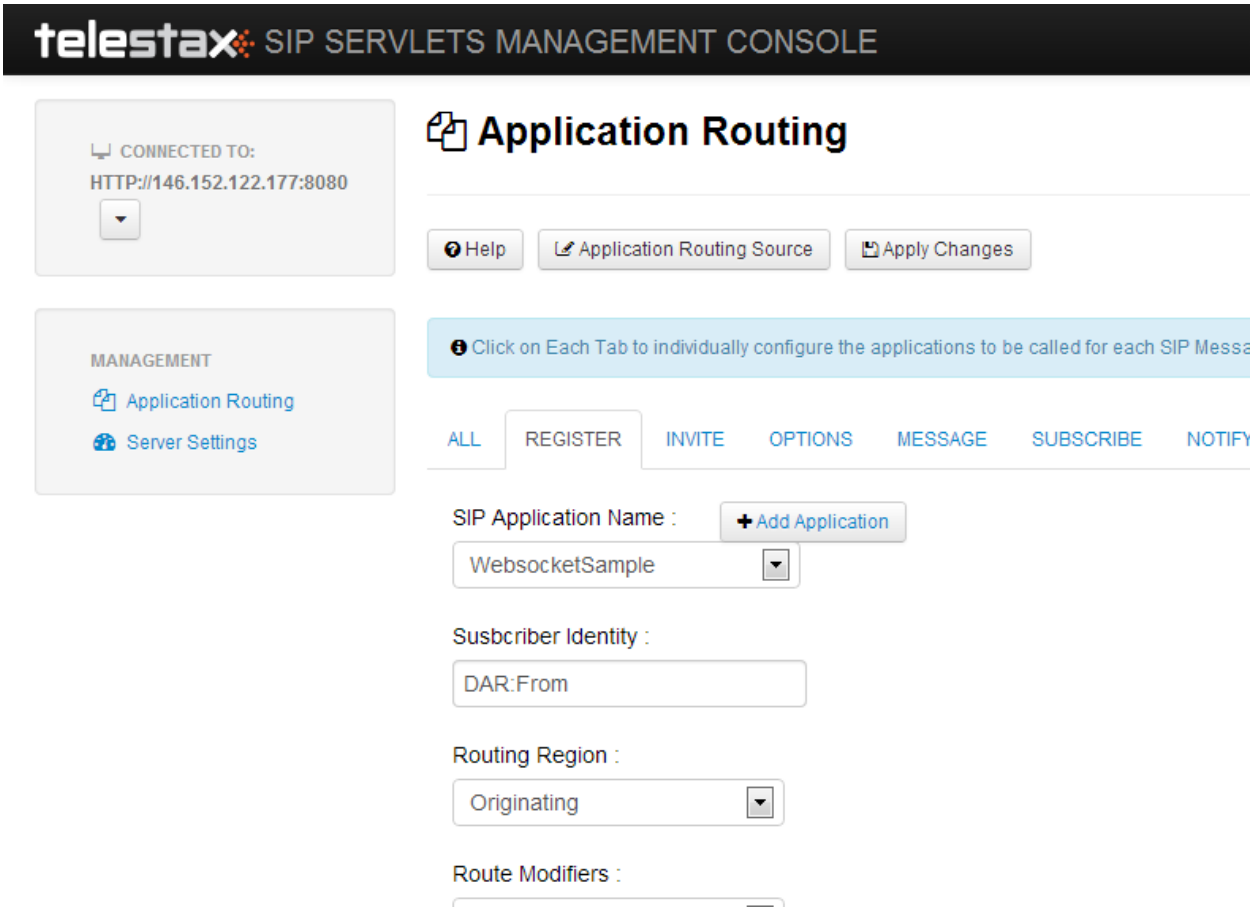
- Click on **Manager App**.



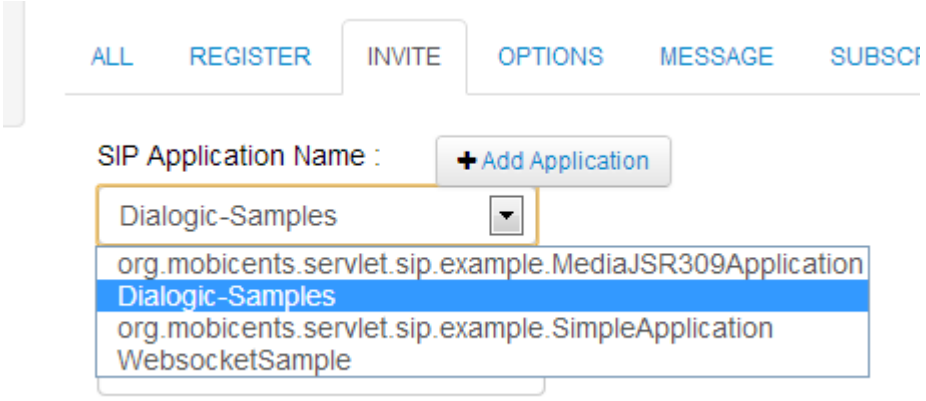
- Put the appropriate credentials.



- Click on **sip-servlets-management**.



- Click on **INVITE** tab and select **Dialogic-Samples** under **SIP Application Name**.



- Click **Apply Changes** button and a message on the bottom of your screen will confirm its success.

```
- [INFO] DAR Information successfully updated
```

- Now, your Application Server is configured to use newly deployed application.

Follow these steps to run ***DlgcPlayerDemo*** for verification:

1. Have a SIP client configured for supported audio codec.
2. Place a call into Application Server with following URI:

```
DlgcPlayerDemo@<as_ip_address>:5080
```

3. With successful configuration, you should hear a verification prompt being played out.

4. Test Servlets

This section describes the test servlets (basic sample applications) and requirements for running test servlets in the JSR 309 Connector.

Test Servlets Overview

Test Servlets, or sample applications, are included as part of distribution. They illustrate the use of the JSR 309 Connector. These test servlets are included in the *dlgmsc_tests.war*. For installation instructions and any additional requirements for running test servlets, refer to Installation and Configuration of JSR 309 Connector Demo and to Proper Configuration of PowerMedia XMS.

Running the Test Servlets

When using any standard SIP phone a special SIP URI will be used to initiate each test servlet.

DlgcPlayerTest

This test servlet plays a PowerMedia XMS pre-set prompt.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcPlayerDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

Using the demo property file, set the following:

```
player.test.prompt=
```

For example:

```
player.test.prompt=file:////var/lib/xms/media/en_US/verification/greeting.wav
```

The player will play this prompt. Make sure that the prompt file exists in the Media Server.

To test the application, dial the following:

```
DlgcPlayerDemo@<as_ip_address>:5080
```

DlgcDtmfPromptAndCollectTest

This test servlet plays a prompt and collects DTMF digits.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcPromptCollectDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

The ***DlgcPromptCollectDemo*** can be controlled using the demo property file as follows:

- The detectOnlyTest reads the number of signals property value and sends the pattern x (times number of signals). Note that no prompt is played. The following example generates a pattern to match of any five (5) DTMF entries:

  ```
  signalDetector.test=detectOnlyTest
  signalDetector.number_of_signals=5
  ```

- The detectPromptCollectTest plays a prompt and looks for a given pattern. It does not make use of the number of signals property.

  ```
  signalDetector.test=detectPromptCollectTest
  signalDetector.match_pattern=min=1;max=5;rtk=#
  ```

- The detectCollectWithPatternTest does not prompt the user and only uses the match_pattern.

```
signalDetector.test=detectCollectWithPatternTest
signalDetector.match_pattern=min=1;max=5;rtk=#
```

Note 1: You can configure the signal detector with the following properties (for example, the timeout values are based in milliseconds units):

```
signalDetector.initial_digit_timeout=5000
signalDetector.inter digit timeout=5000
signalDetector.max_duration=10000
```

Note 2: For the test that plays a prompt, you can control a loop (or how many times the test repeats the prompt and collect) by controlling the following property:

```
signalDetector.loopCounter=2
```

To test the application, dial the following:

```
DlgcPromptCollectDemo@<as_ip_address>:5080
```

DlgcRecorderTest

This test servlet records a greeting.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcRecorderDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created test contact. You are prompted to record your greeting at the tone. After the tone, say your greeting, and enter **#000** to play your greeting.

After the greeting is played back, the application completes by hanging up the phone. If you do not enter **#000**, the greeting continues to record until the timeout is reached.

The recording demo can be controlled in the demo property file by configuring the following record properties:

```
record.test.file=file:////tmp/recorder jsr309 test demo.ulaw
record.test.minDuration=6000
record.test.maxDuration=60000
record.test.initialTimeout=7000
record.test.finalTimeout=4000
record.test.silenceTerminationFlag=true
```

To test the application, dial the following:

```
DlgcRecorderDemo@<as_ip_address>:5080
```

DlgcDtmfAsyncTest

This test servlet illustrates the asynchronous DTMF capabilities.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcAsyncDtmfDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created test contact. Notice that there are no prompts. You will be connected. The application waits for you to press DTMF digits. For each DTMF pressed, the application will receive the DTMF and print the collected DTMF to the screen.

Selecting the number **0** hangs up the connection.

To test the application, dial the following:

```
DlgcAsyncDtmfDemo@<as_ip_address>:5080
```

Conference Demos

The following table depicts the conference demos that are delivered with JSR 309 Connector:

Demo Name	Functionality	Requires
JMCConferenceServlet	Demonstrates how to create and manage multiple conferences.	Media files need to be installed in the PowerMedia XMS for menu to work.
DlgcAvLayoutConferenceDemo	Implements an advanced conference.	Media files need to be installed in the PowerMedia XMS for menu to work. The demo property file must be configured.
DialogicBridgeConference	Shows how to create a two leg conference without using a mixer.	Media files need to be installed in the PowerMedia XMS for menu to work.

JMCConferenceServlet

This test servlet illustrates how to create and manage multiple conferences using a mixer control leg. A mixer control leg is an extra SIP connection used to control the conference mixer.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcMultiConferenceDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This conference performs the following:

1. Establishes a network connection and joins it with a media group.
2. Plays a prompt for a new number (conference pin) and collects signals. Any pin number can be provided. Initially no conferences exist. Conferences are created as users call in and provide pin numbers. Callers will only hear other callers who provide the same pin number.
3. Creates a conference if a new pin is used, or adds a leg to an existing conference.

To test the application, dial the following:

```
DlgcMultiConferenceDemo@<as_ip_address>:5080
```

DlgcReferenceConferenceWithOutBCallServlet

This test servlet illustrates how to implement an advanced conference that does not require a mixer control leg. The legs are connected directly into a conference without requiring any initial IVR functionality.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcAvLayoutConferenceDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This simple conference performs the following:

1. Can join multiple legs into a conference.
2. Once in conference, the user can enter ***00** to hear the conference menu and apply some of the menu options.

The demo can be controlled by configuring the following properties in the demo property file:

- Change the initial direction of legs by entering the following properties in the application demo property file:

```
demos.join.direction.leg1=<duplex,recv,send>
demos.join.direction.leg1=<duplex,recv,send>
demos.join.direction.leg1=<duplex,recv,send>
```

- To make an outbound call, make sure you have another accessible SIP phone that can receive calls and configure the following attributes:

```
application.sipTOA_Address.sip.address=146.152.245.3 # IP address of the SIP Phone
application.sipTOA_Port.sip.port=5060
application.sipTOA.sip.username=kapanga  #(any name will do)
application.early media bridge.sip.address=146.152.122.127 #AS Addr
application.early_media_bridge.sip.port=5060 #AS SIP PORT
```

- To run a video conference, make sure you set the following configuration:

```
media.mixer.mode=AUDIO_VIDEO  # possible values AUDIO,AUDIO_VIDEO
media.mixer.conf.video.size=VGA  # possible values VGA, 720p
media.mixer.conf.recordfile=file:////tmp/confRecording  # recording the conference file
full path. This also works for audio only conference.
```

  Note: To play the conference recording after the recording is completed, change the following attribute to point to the recording path:

```
player.test.prompt=file:////tmp/confRecording then run DlgcPlayerDemo
```

To test the application, dial the following:

```
DlgcAvLayoutConferenceDemo@<as_ip_address>:5080
```

DialogicBridgeConference

This test servlet illustrates how to implement a simple conference that does not require a mixer, and that has two legs directly joined into it.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcBridgeDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This simple conference performs the following:

- Joins two calling legs into a simple conference.
- In order for the leg to enter the bridge, each leg must enter ***03** after making the call.

To test the application, dial the following:

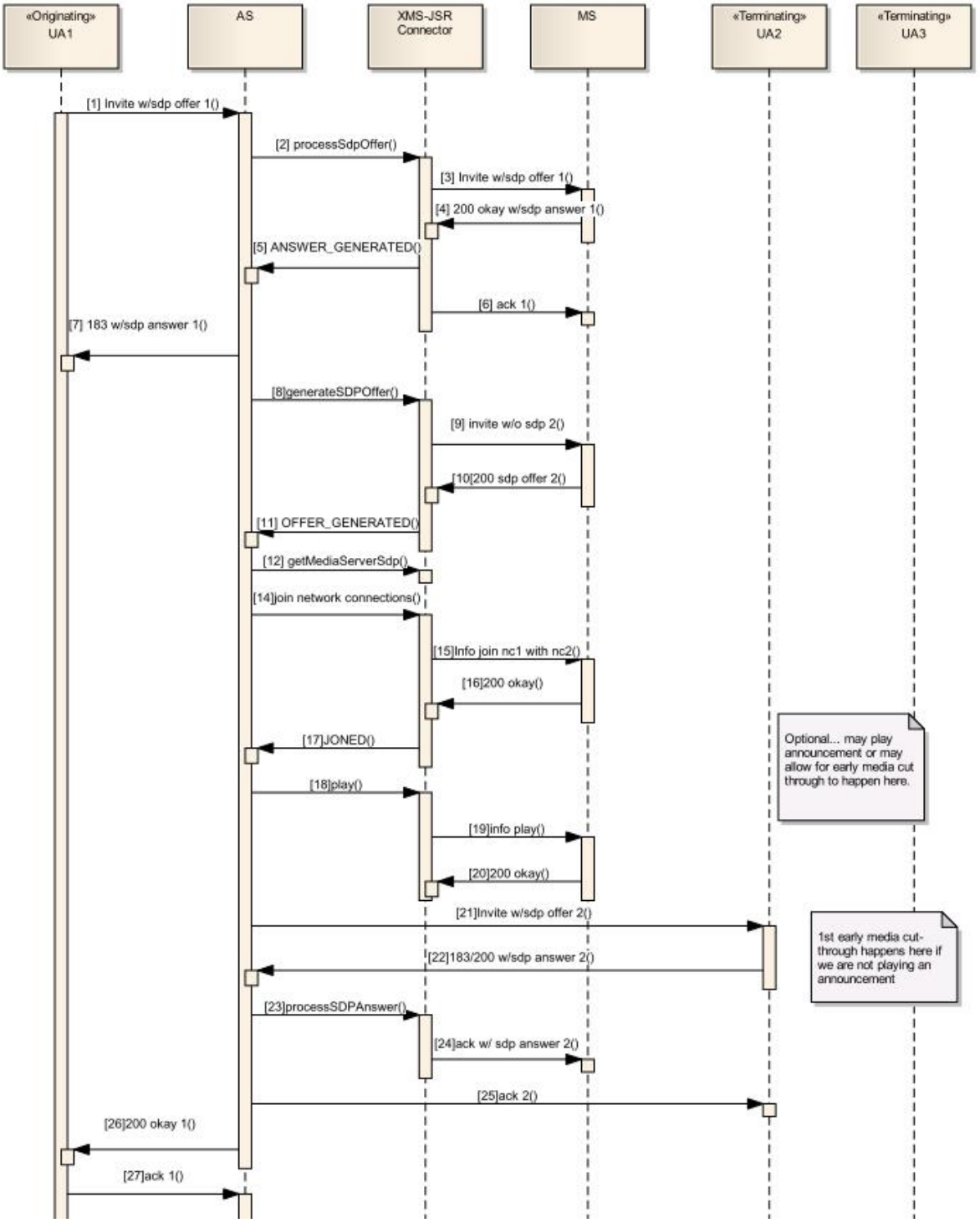
```
DlgcBridgeDemo@<as_ip_address>:5080
```

DlgcEarlyMediaBridgeDemo

This test servlet is similar to the DialogicBridgeConference defined above, except that it simulates an early media scenario.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to ***DlgcEarlyMediaBridgeDemo***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

The following sequence diagram illustrates ***DlgcEarlyMediaBridgeDemo***:


«Originating»
UA1
AS
XMS-JSR
Connector
MS
«Terminating»
UA2
«Terminating»
UA3
[1] Invite w/sdp offer 1()
[2] processSdpOffer()
[3] Invite w/sdp offer 1()
[4] 200 okay w/sdp answer 1()
[5] ANSWER_GENERATED()
[6] ack 1()
[7] 183 w/sdp answer 1()
[8]generateSDPOffer()
[9] invite w/o sdp 2()
[10[200 sdp offer 2()
[11] OFFER_GENERATED()
[12] getMediaServerSdp()
[14]join network connections()
[15]Info join nc1 with nc2()
[16]200 okay()
[17]JONED()
Optional... may play announcement or may allow for early media cut through to happen here.
[18]play()
[19]info play()
[20]200 okay()
[21]Invite w/sdp offer 2()
1st early media cut-through happens here if we are not playing an announcement
[22]183/200 w/sdp answer 2()
[23]processSDPAnswer()
[24]ack w/ sdp answer 2()
[25]ack 2()
[26]200 okay 1()
[27]ack 1()

Menu supported by ***DlgcEarlyMediaBridgeDemo***:

***00** – Plays announcement of menu options

***77** – Plays announcement of how the demo works

***88** – Plays announcement informing the user if the application is in a bridge or mixer conference

***99** – Transfers the two call leg from a bridge conference to a full conference using a mixer

Note: Once in a mixer conference, the test application does not allow you to go back to a bridge conference. The following property configuration must be set for this demo to work:

```
application.sipTOA Address.sip.address=146.152.245.3 # IP address of the SIP phone
application.sipTOA Port.sip.port=5060
application.sipTOA.sip.username=kapanga  #(any name will do)
application.early media bridge.sip.address=146.152.122.127 #AS Addr
application.early_media_bridge.sip.port=5060 #AS SIP PORT
```

To test the application, dial the following:

```
DlgcEarlyMediaBridgeDemo@<as_ip_address>:5080
```

5. Troubleshooting

This section provides basic troubleshooting techniques for the JSR 309 Connector.

Logging

The JSR 309 Connector and its sample applications use the Apache Log4j 2 logging facility. Unlike the previous version, the connector makes use of *log4j2.xml* file for its logging configuration. This *log4j2.xml* configuration file needs to be part of applications WAR file under "*WEB-INF\classes*" directory. With the introduction of Log4j 2, it is possible to change the log levels without stopping/restarting any components. All the user needs to do is open *log4j2.xml* and change the logging to desired level.

The JSR 309 Connector and sample applications log output file can be found in the "*$CATALINA_HOME/logs/*" directory - *dlgmsc.log*. The default logging level is set to debug.

The application can also take an advantage of using Log4j 2 facility where it will have to make changes to the *log4j2.xml* file and accordingly add the appropriate appenders and loggers. Refer to the Apache Log4j 2 documentation at http://logging.apache.org/log4j/2.x for details.

The application can also utilize platform logging facility where its configuration and modifications can be accomplished via appropriate Application Server Administration page. Refer to the application specific documentation for details.

SIP Errors

If the PowerMedia XMS returns "503 Service Unavailable", make sure your network is correctly set up by performing the following actions:

1. Verify the available PowerMedia XMS licenses.
2. Check the "*/etc/hosts*" file configuration.
3. Make sure application properties file (i.e., *dlgc_demos.properties*) is referencing the appropriate PowerMedia XMS and Application Server IP address and ports.

6. Building and Debugging Sample Demos in Eclipse IDE

The JSR 309 Connector distribution comes with necessary configuration files and content needed to build Dialogic sample applications. This section is going to provide the steps on how to create, compile, build, and debug provided demo application using Eclipse IDE.

Prerequisites

User will need to have installed the following components:

- Latest JDK version 1.7.

  Note 1: JDK 1.7.0_60 was used at the time of this publication.

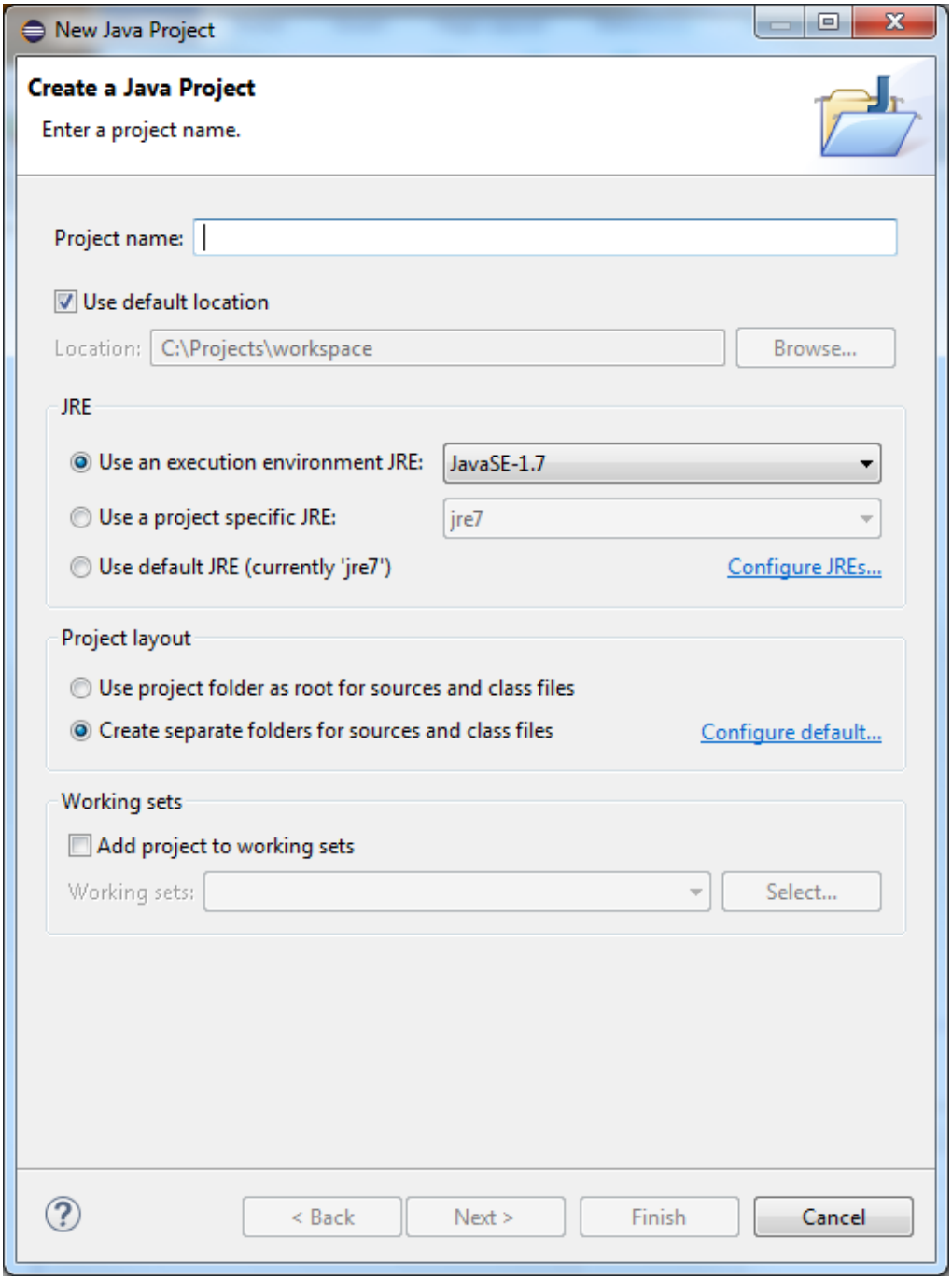
  Note 2: Latest version of 1.7 JDK should be used as this is a version supported by TeleStax Application Server.

- Eclipse KDE (Eclipse Standard SDK – Kepler Service Release 2 used here).
- In order to build provided demo applications, you will need to obtain two TeleStax platform dependent libraries which are NOT provided as part of JSR 309 Connector distribution package. They can be found under the "*CATALINA_HOME/lib*" directory:
  - (TelScale)
    - *sip-servlets-spec-7.0.2.GA-TelScale.jar*
    - *servlet-api.jar*
  - (Mobicents)
    - *sip-servlets-spec-3.0.xxx.jar (xxx represents version of Mobicents 536 for example)*
    - *servlet-api.jar*

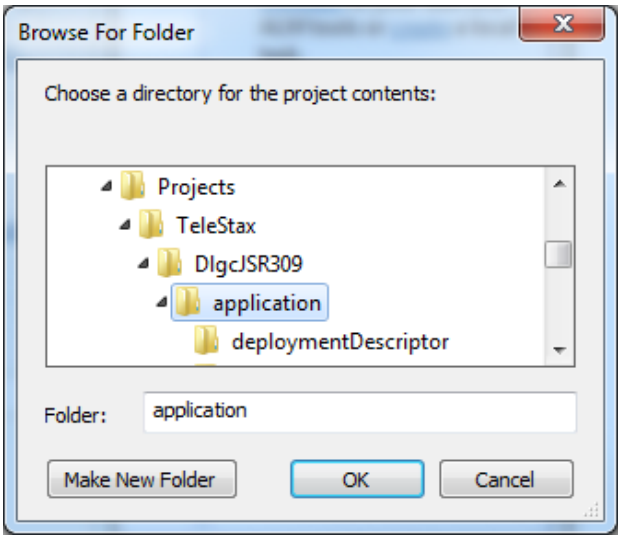
Creating Build Environment

Follow these steps to create a Dialogic demo build environment:

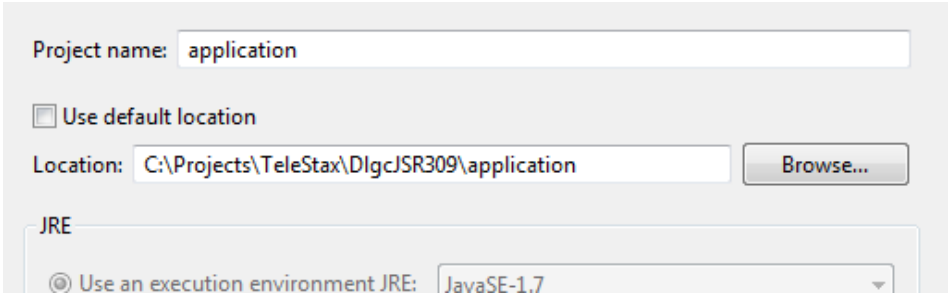
1. From the distribution package, copy the "*DlgcJSR309*" directory and its content to a known location on your system.
2. Copy the required Application Server Platform specific libraries into the "*DlgcJSR309/lib*" directory.
3. Open **Eclipse IDE** and go to **File > New > Java Project**. The following window will appear:



4. Uncheck **Use default location** and then click on **Browse** button.



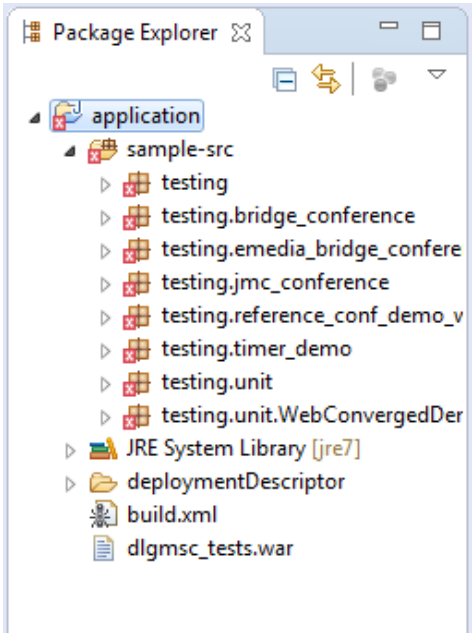
5. Browse to the location of the copied "*DIgcJSR309*" directory and select the **application** directory. Then, click **OK**.



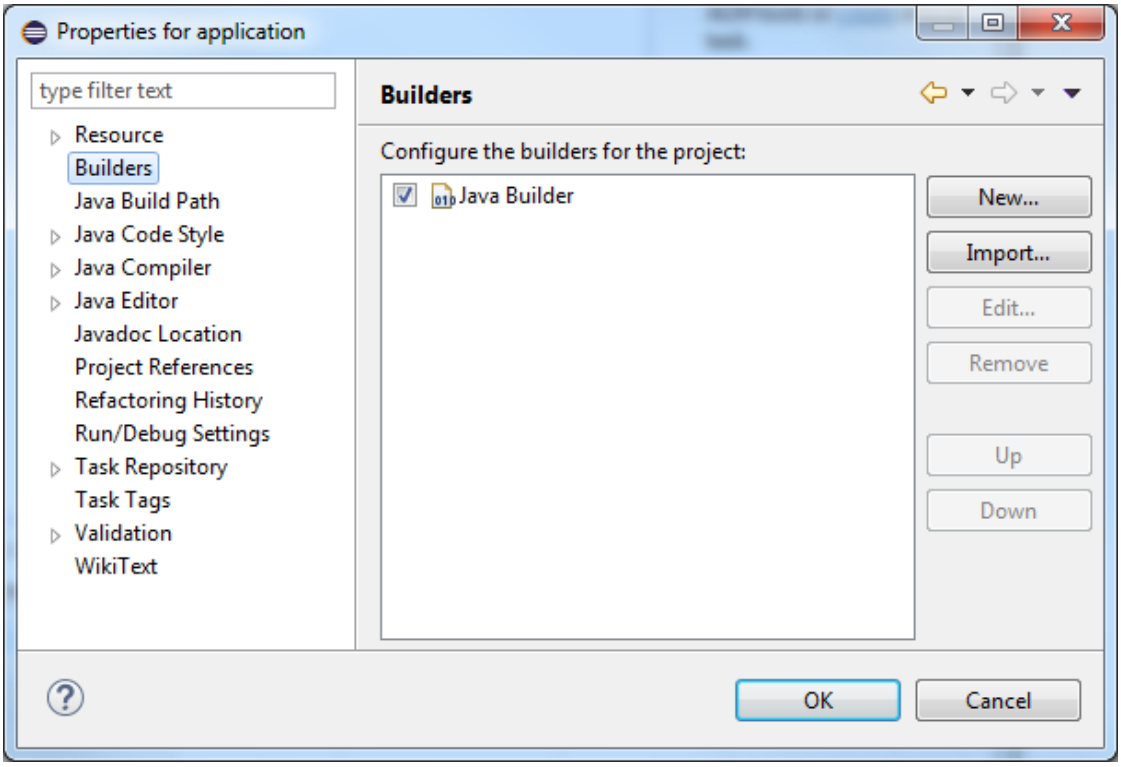
Note: Enter any **Project name** you wish to use in the **Project name** field.

6. Now, click on **Finish** button.

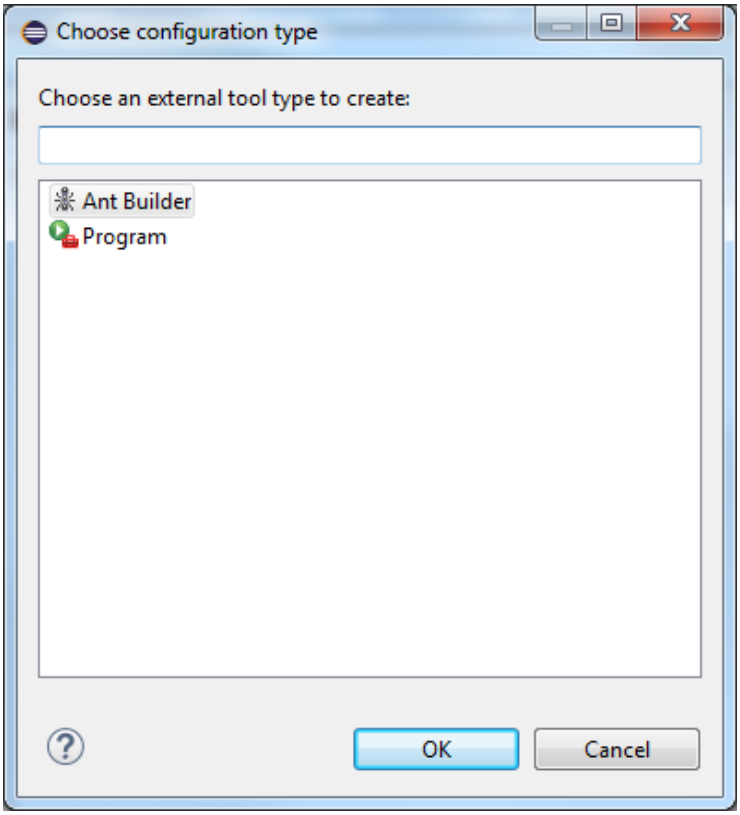
Expanding the project in Eclipse should give you the following:



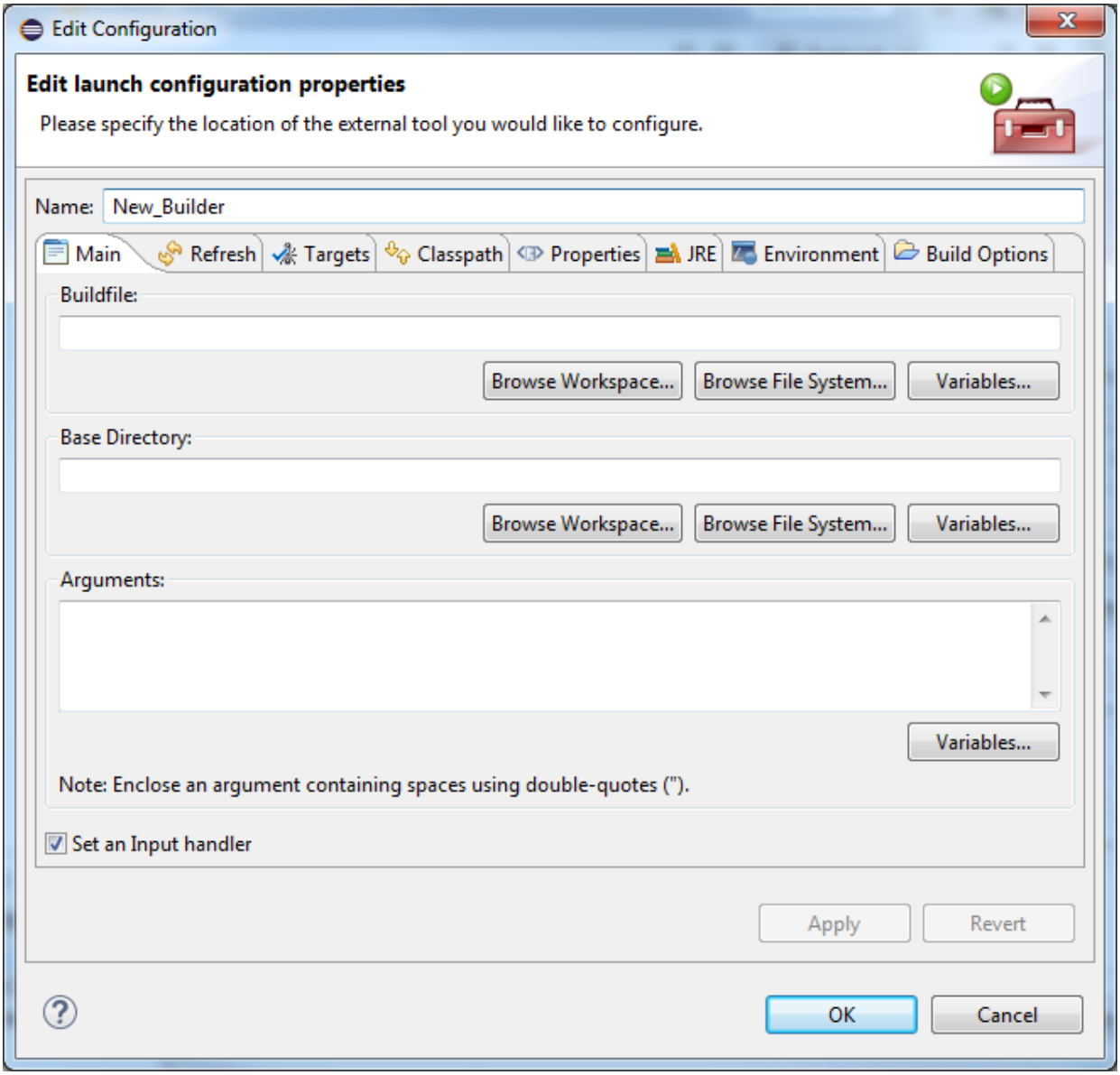
Next, right click on the name of your project in the **Package Explorer** view and select **Properties**.



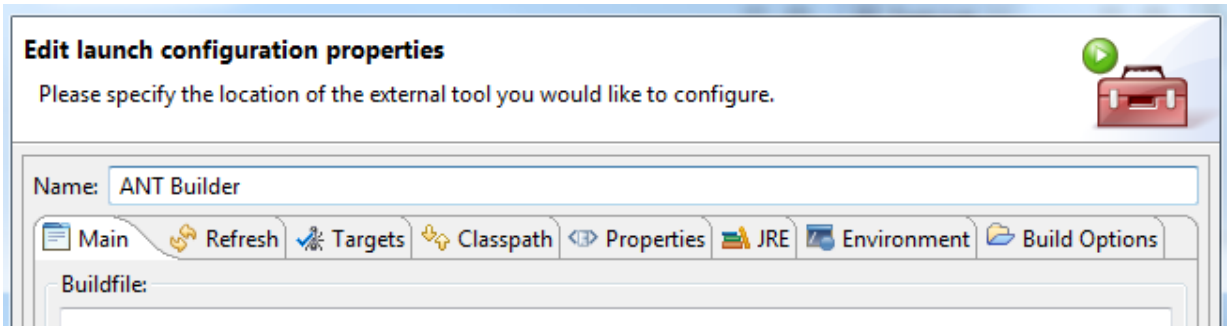
To configure for **ANT builder**, click on **Builders**. Now, deselect the existing **Java Builder** and click on **New** button.



Select **Ant Builder** and then click **OK**.

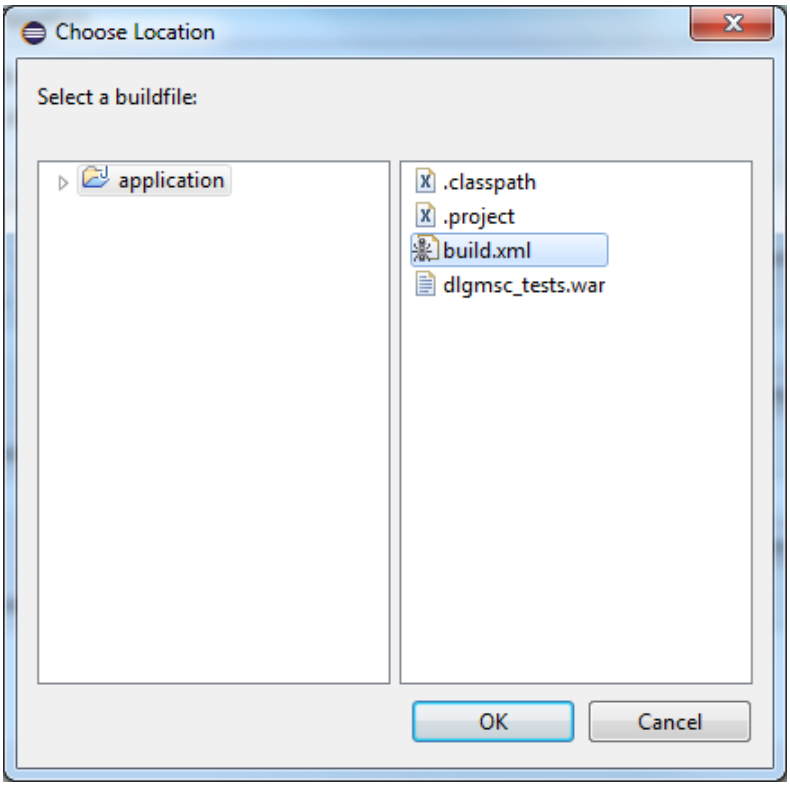


To name the builder, enter in a name in the **Name** section as shown below:

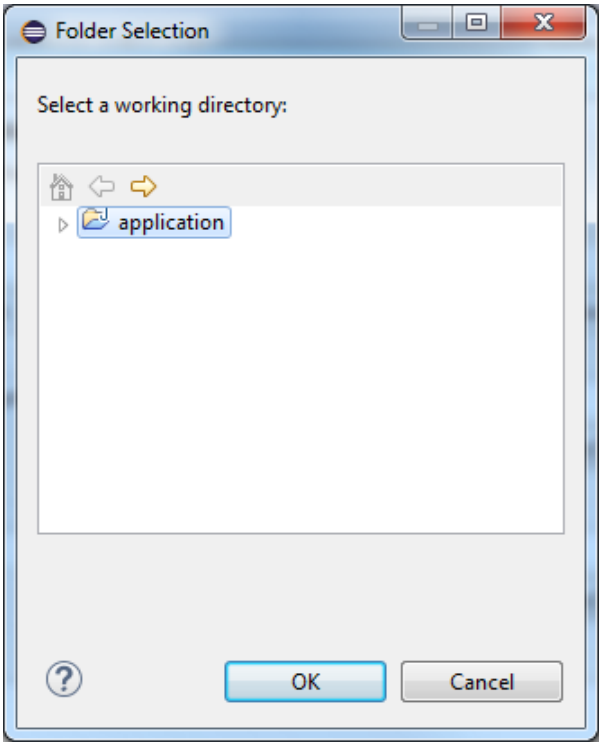


Use the **Main** tab to define **Buildfile** and **Base Directory**:

- Under **Buildfile,** click on **Browse Workspace** button and select the *build.xml* file in the **application** directory. Then, click **OK**:

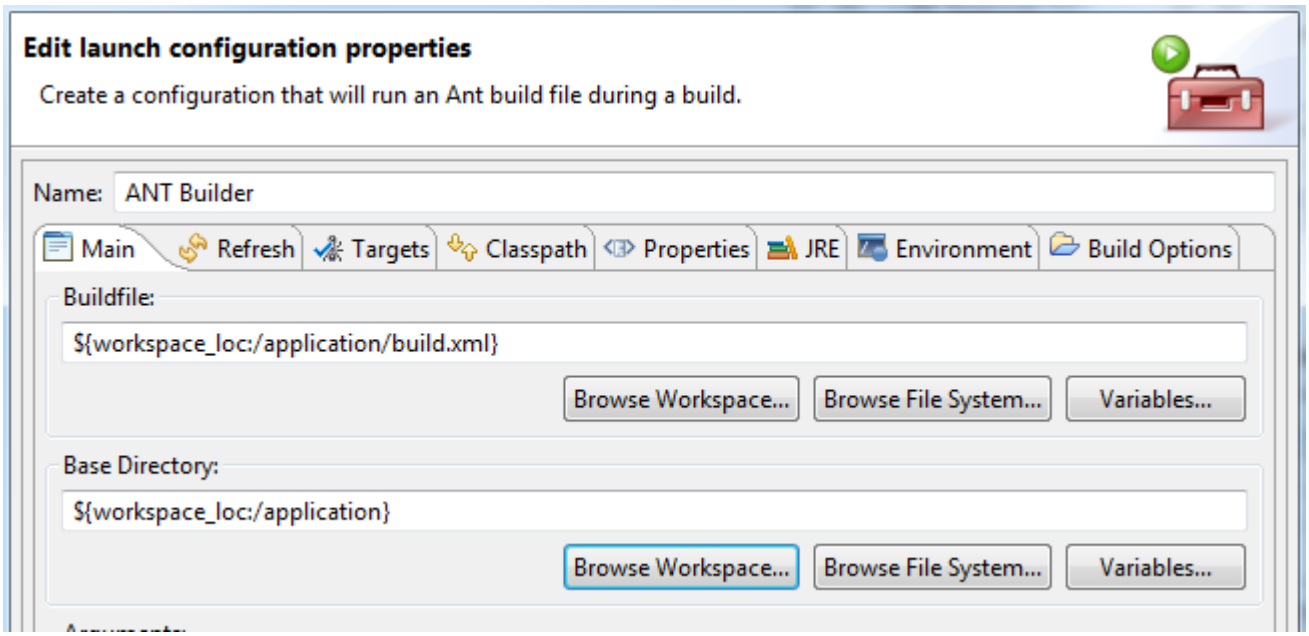


- Next, under **Base Directory**, click on **Browse Workspace** button and select the **application** directory.

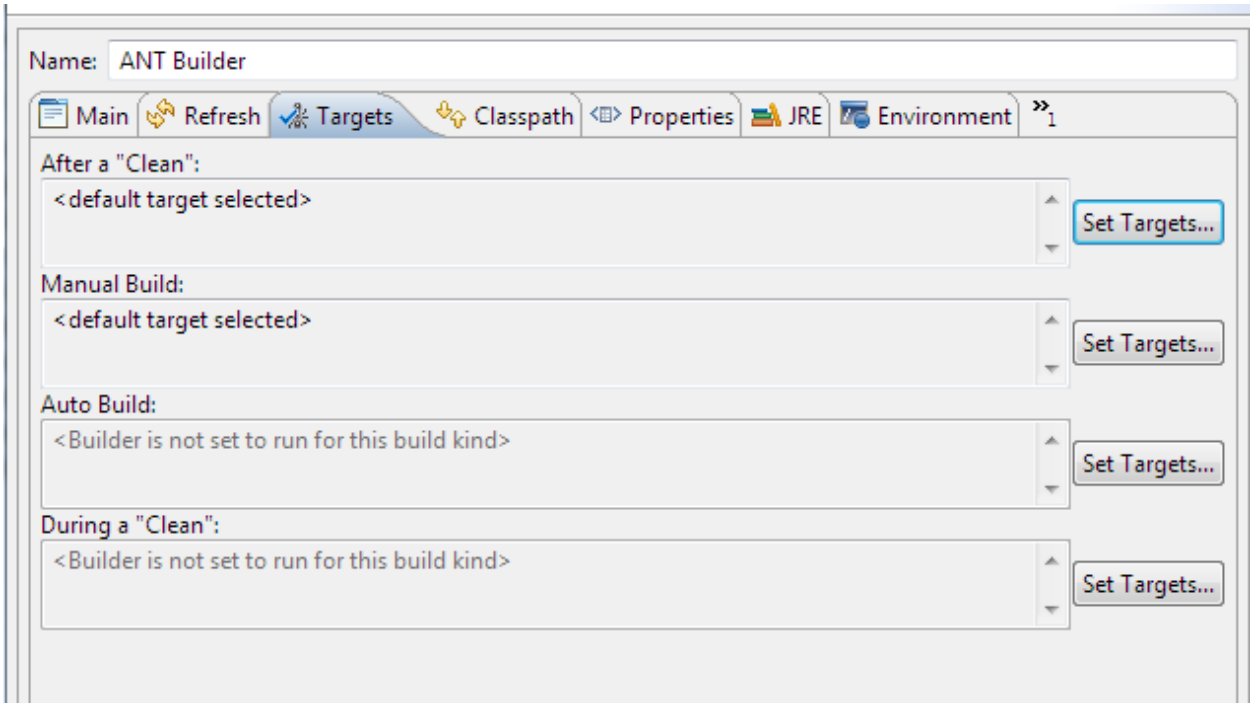


- Then, click **OK**.

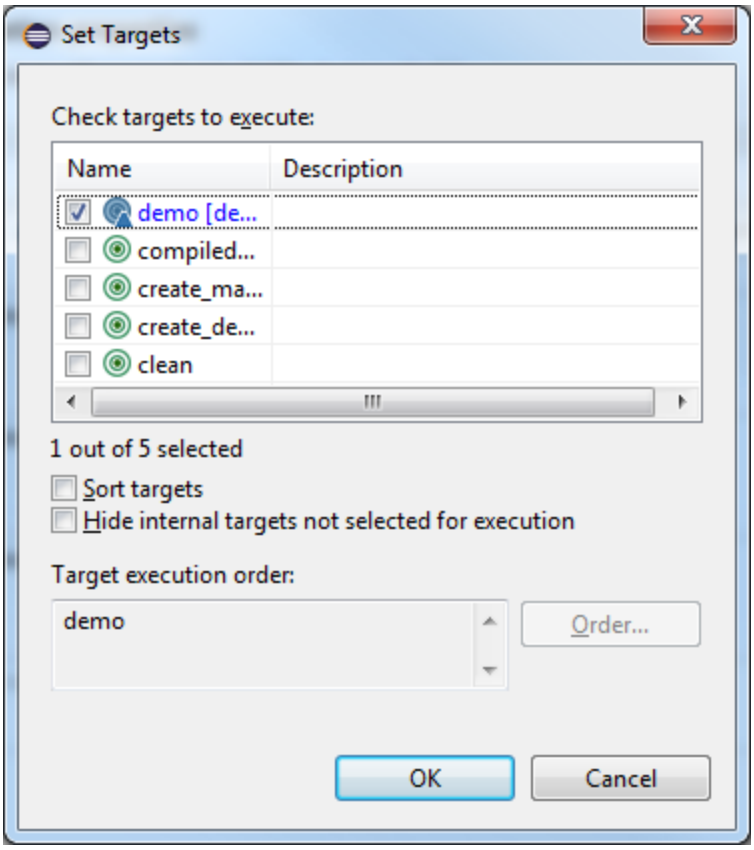
The above changes will reflect the main configuration menu as shown below:



Now, select the **Targets** tab:



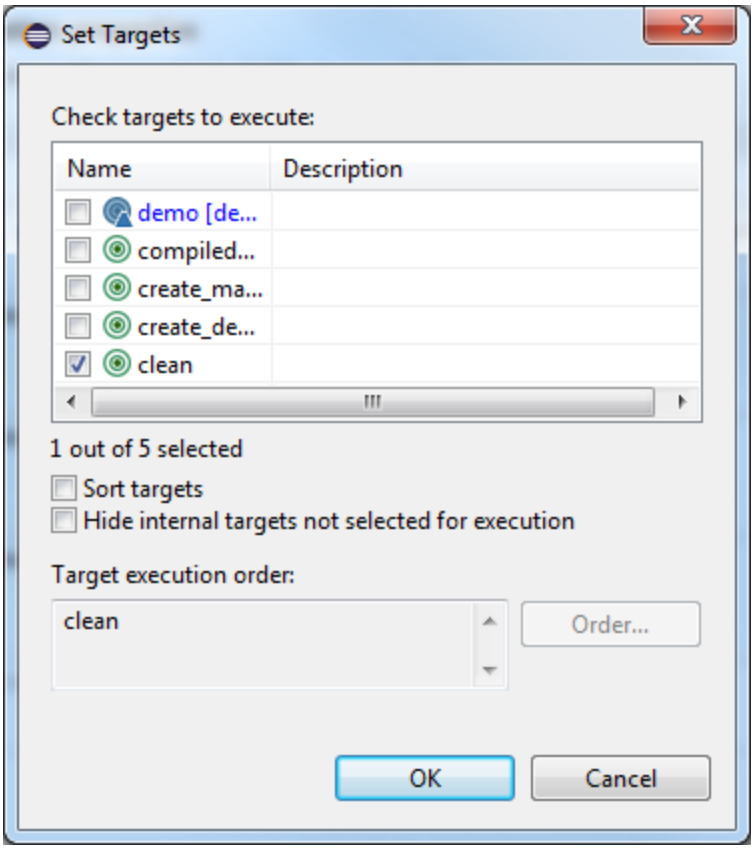
Under **Auto Build,** click the **Set Targets** button. The **demo** target must be selected as illustrated below:



Then, click **OK**.

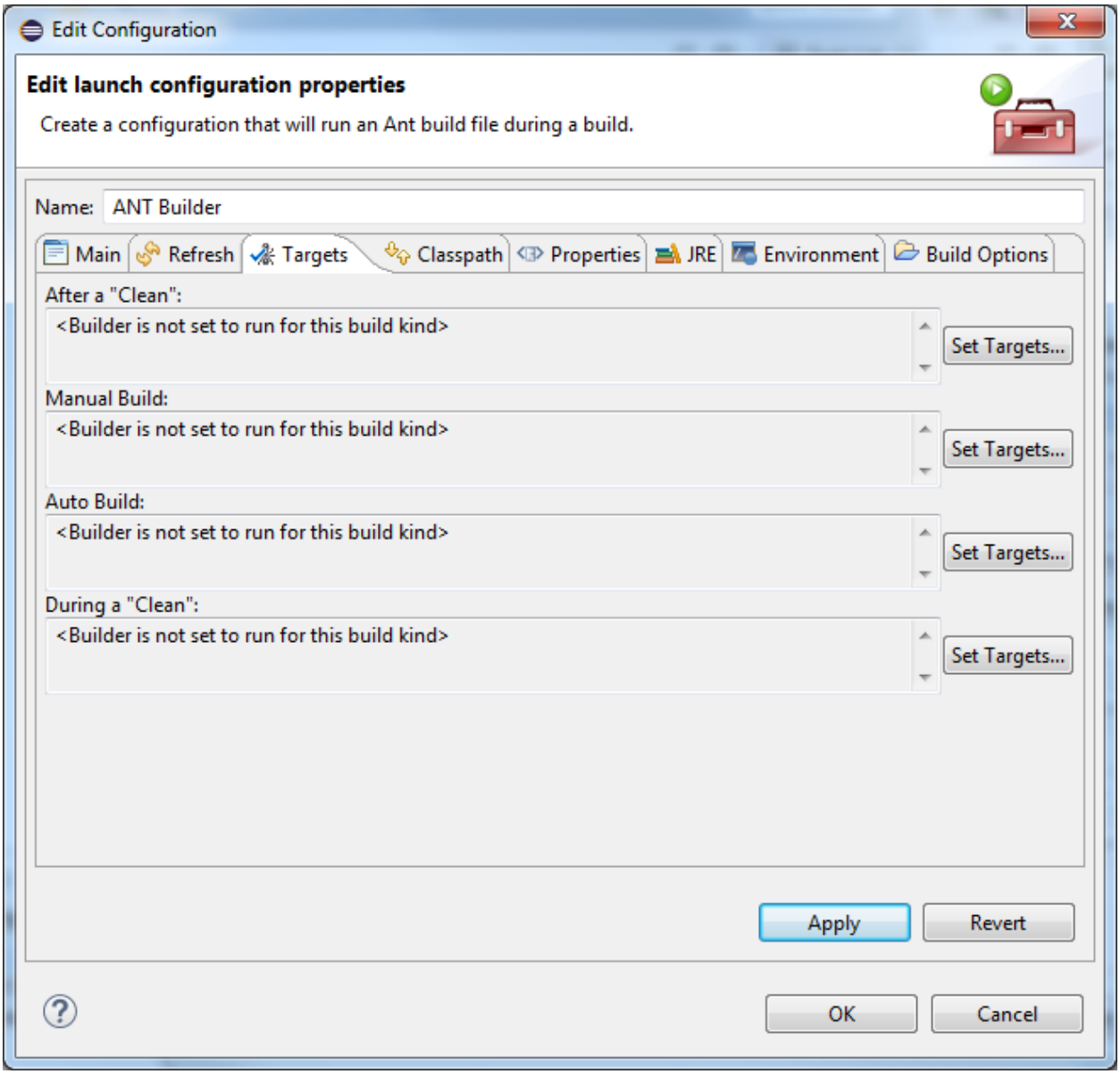
You will now be returned to the **Targets** tab on the main configuration menu. Under **During a "Clean"**, click on **Set Targets** button. The only target that should be selected is the **clean** target.

Note: The **demo** target will most likely be selected by default, in which case you will need to deselect it.

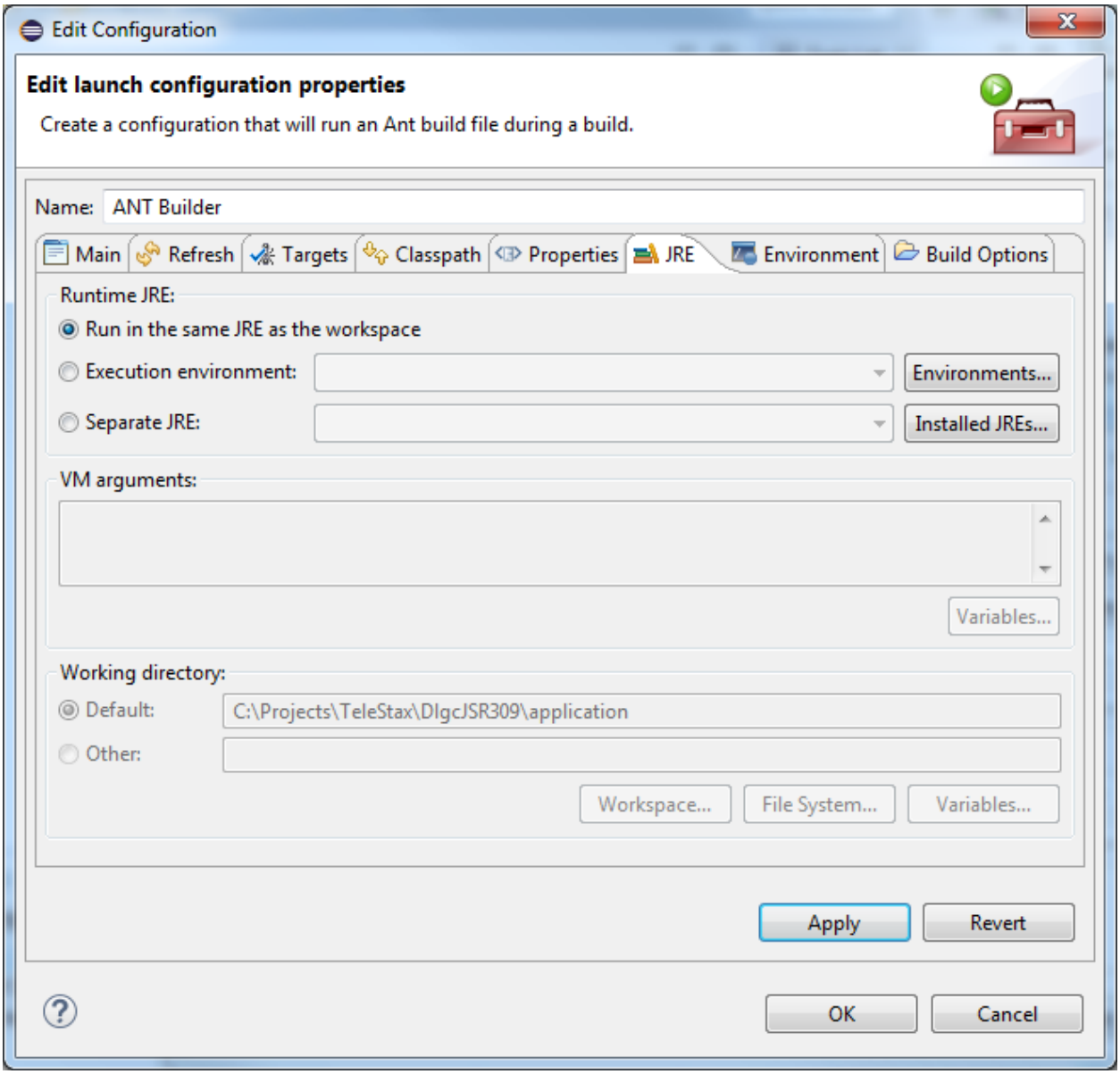


Then, click **OK**.

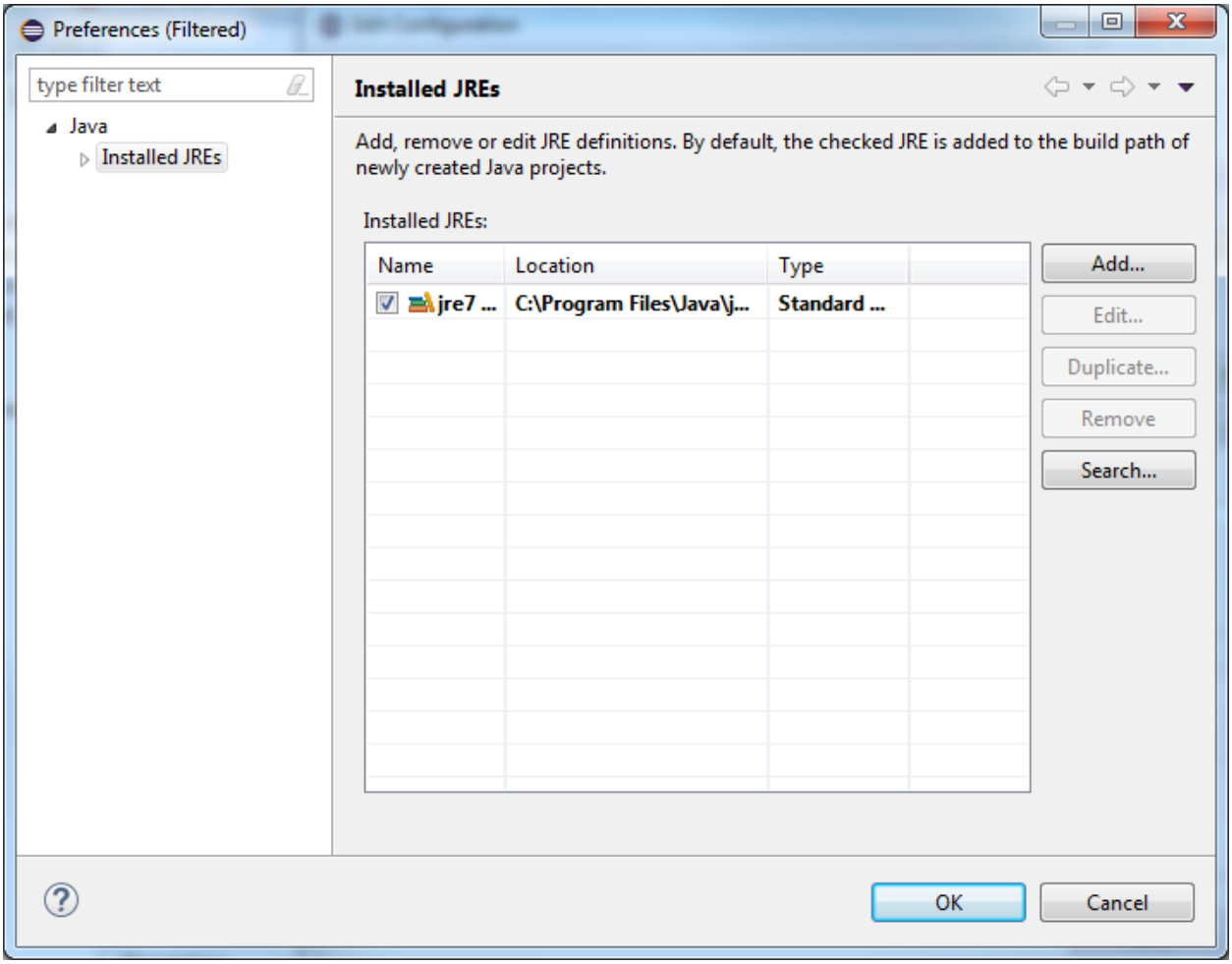
Once returned to the **Targets** tab on the main configuration menu, click on **Apply** button:



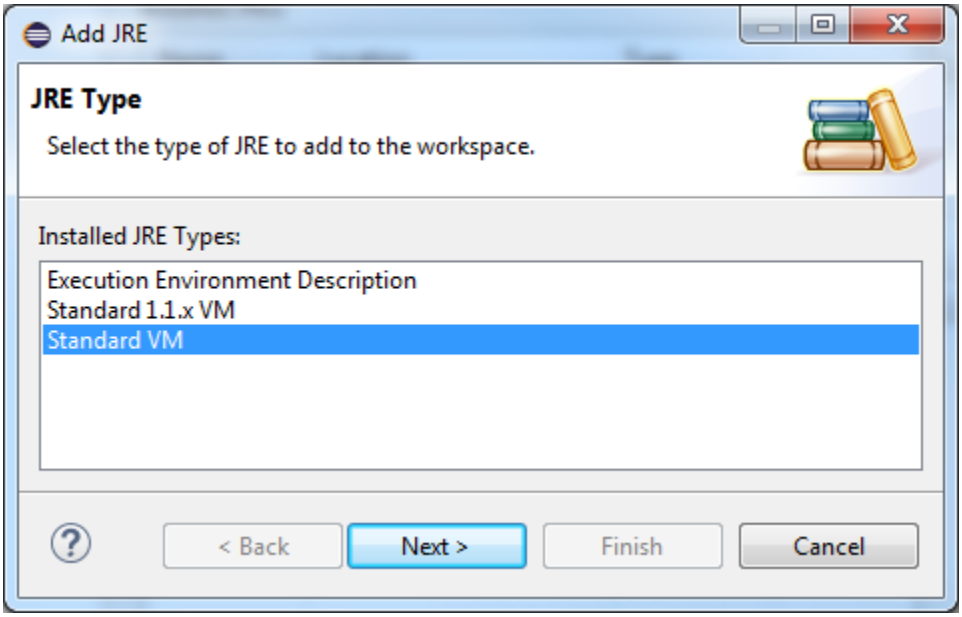
Next, to select the appropriate **JRE environment** which will be used for this project, click on **JRE** tab:



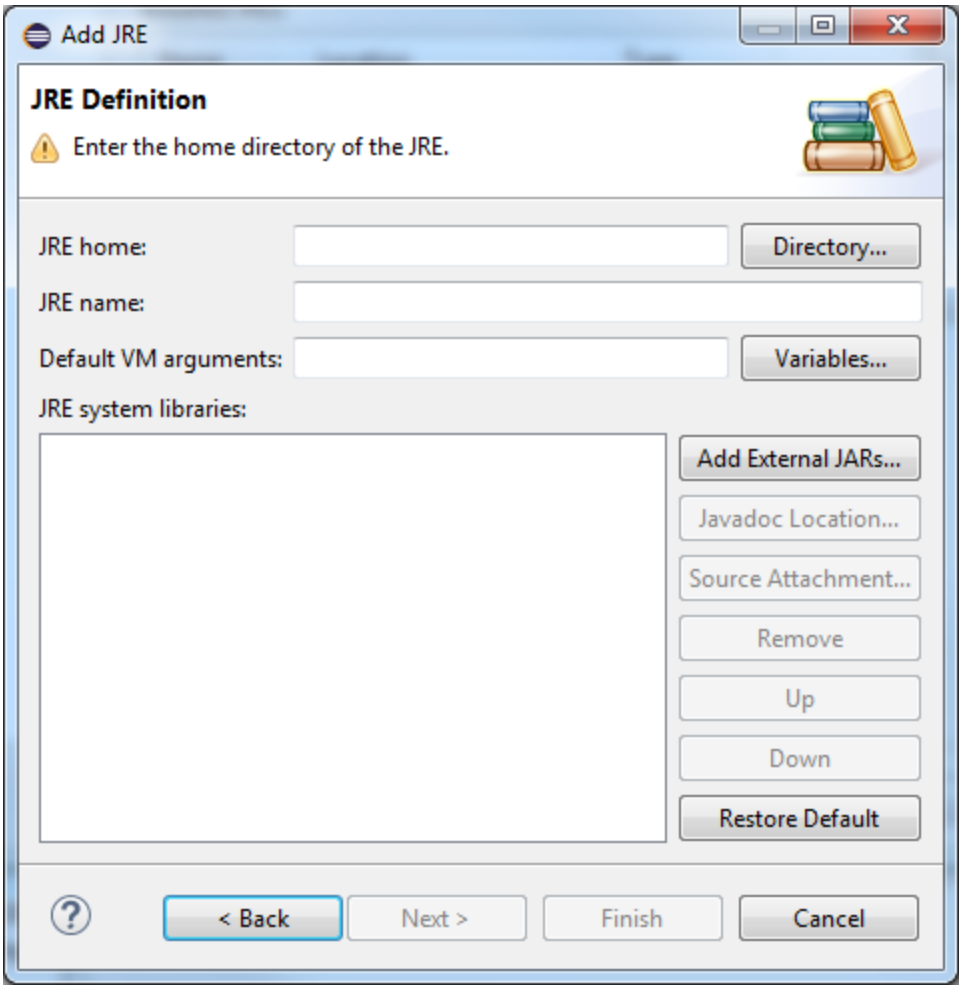
Under **Runtime JRE**, click on **Separate JRE** button. Then, click on **Installed JREs** button:



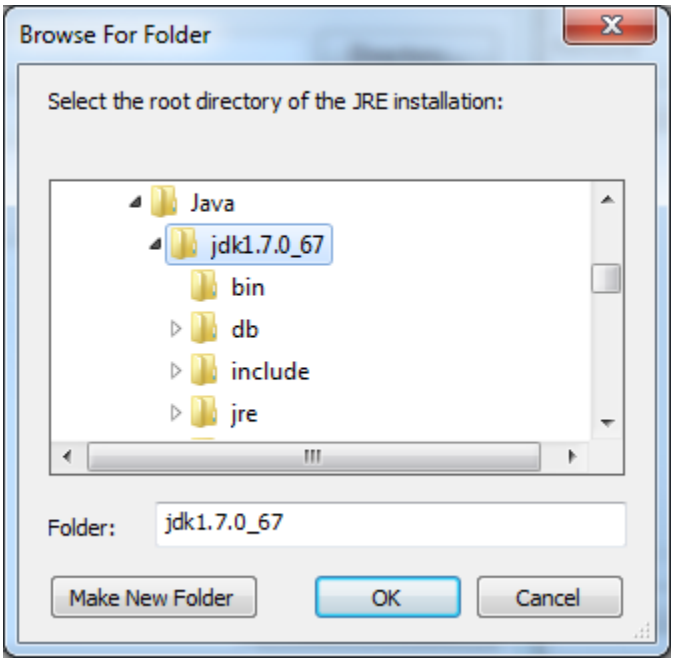
If there is no option under **Installed JREs** for *jdk1.7.0_67* (version based on what was used in this installation example), click **Add**. Then, select **Standard VM**.



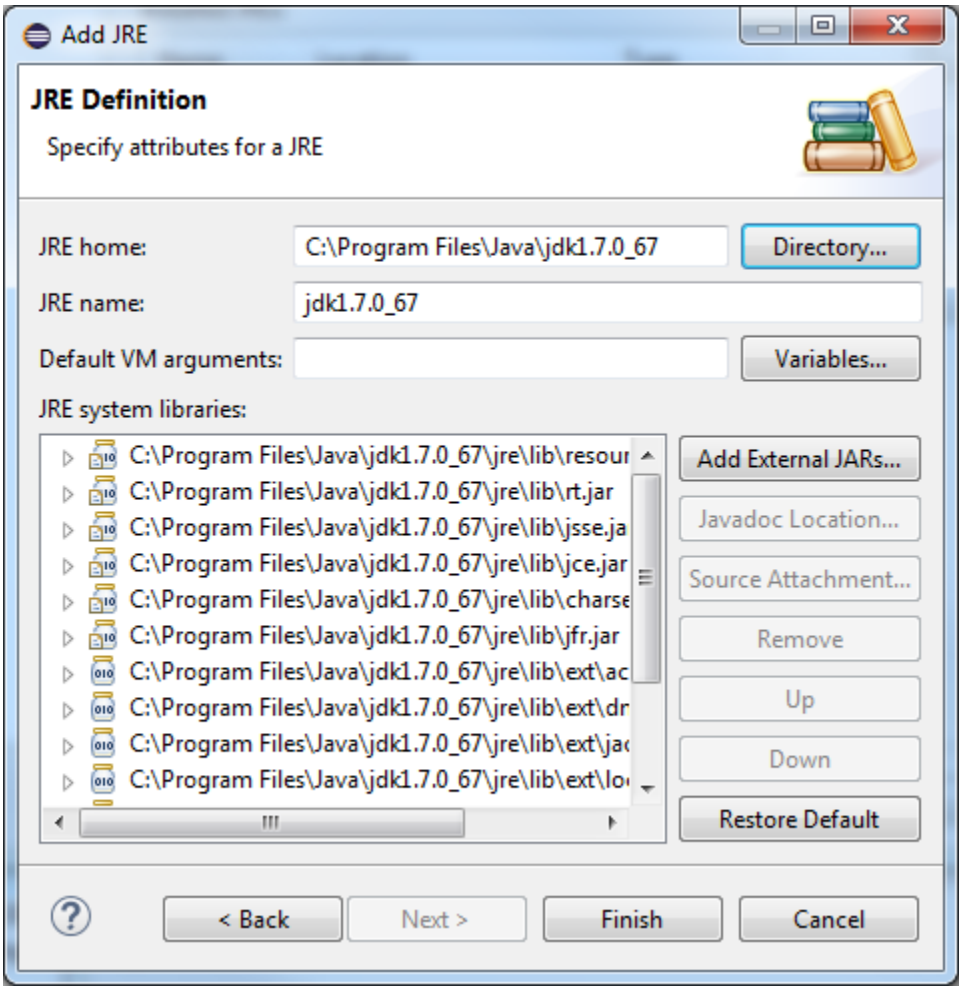
Click **Next**.



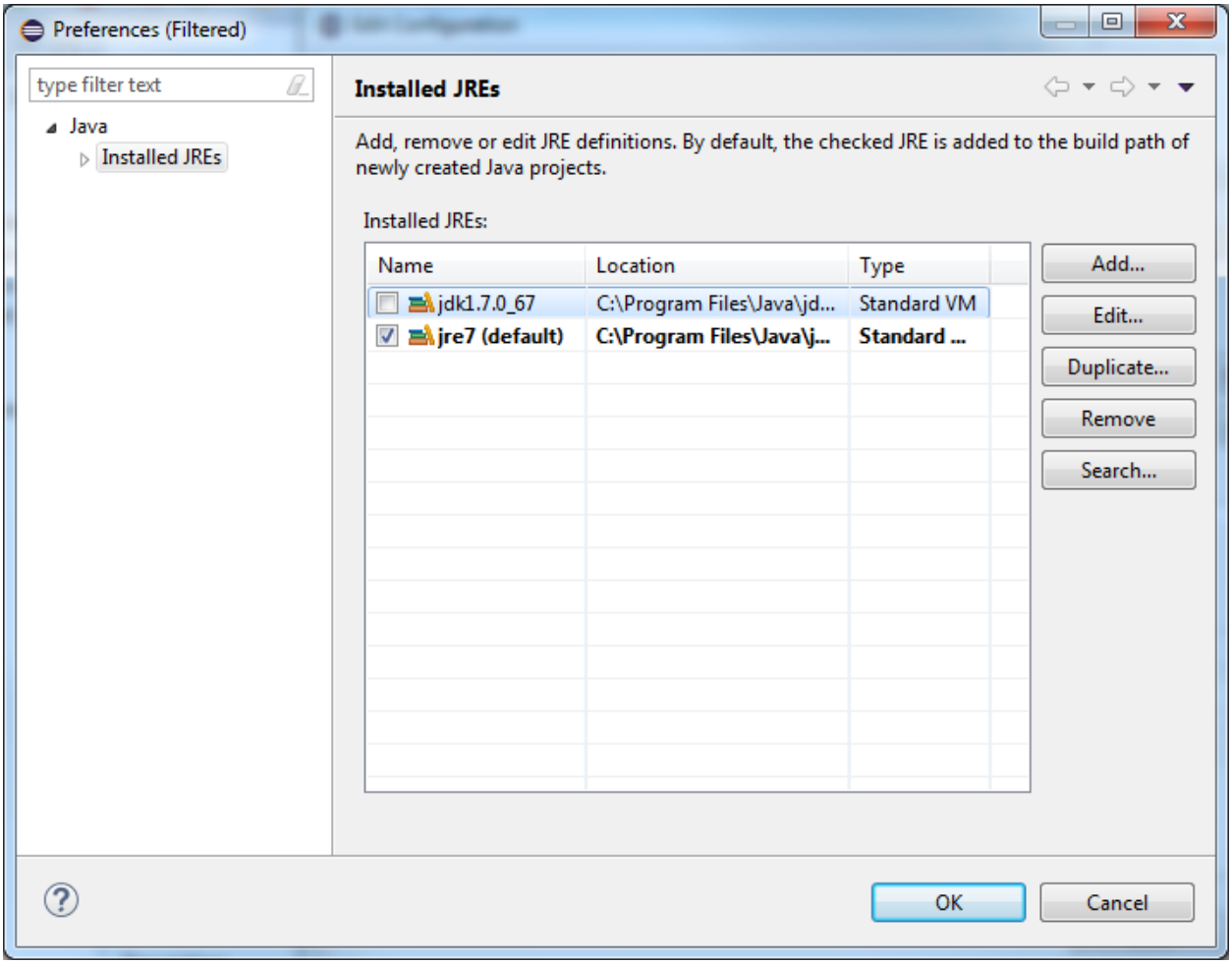
Next, navigate to the location of your installed *jdk1.7.0_67* file by clicking on **Directory** button.



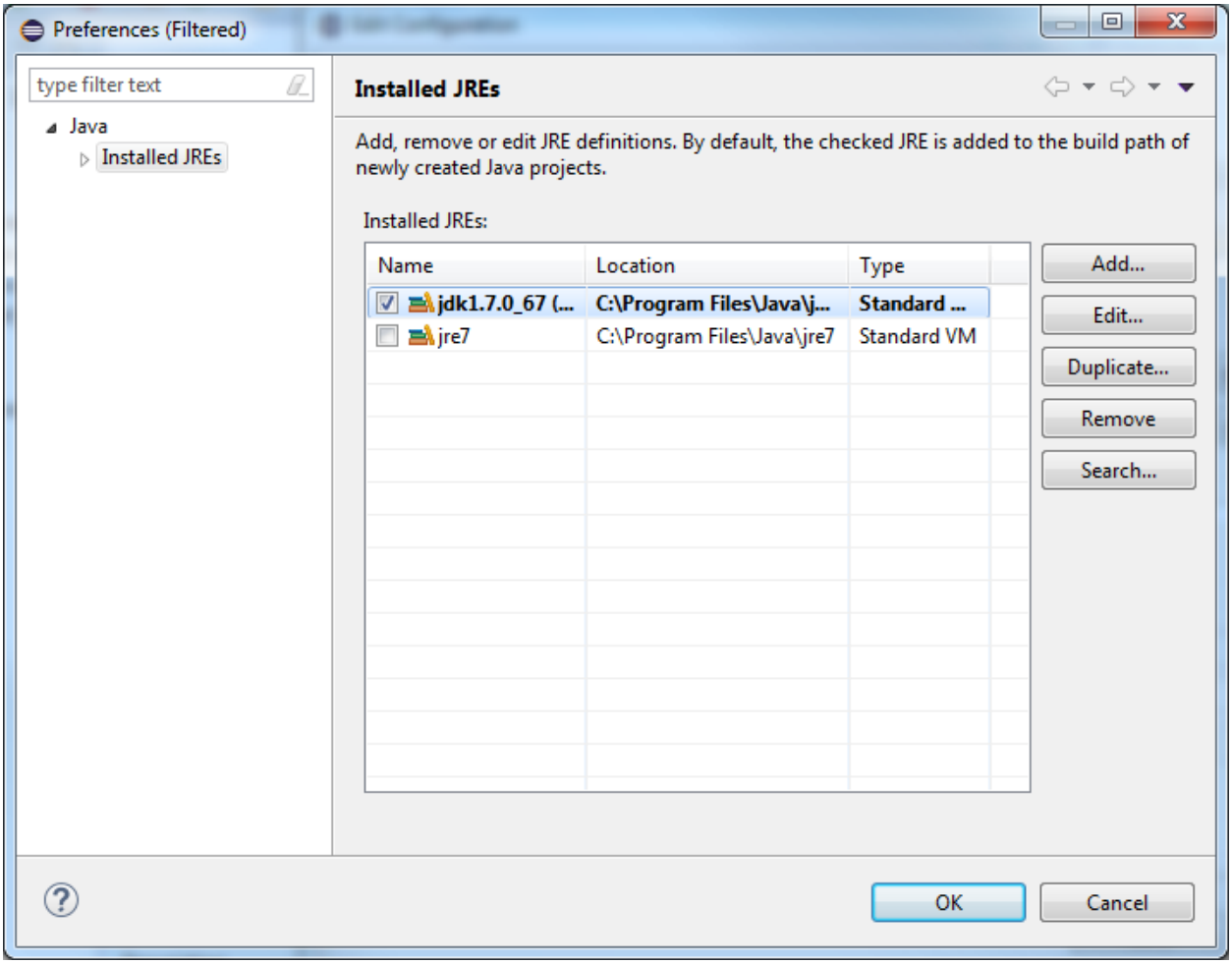
Select the appropriate **JDK** and click **OK**:



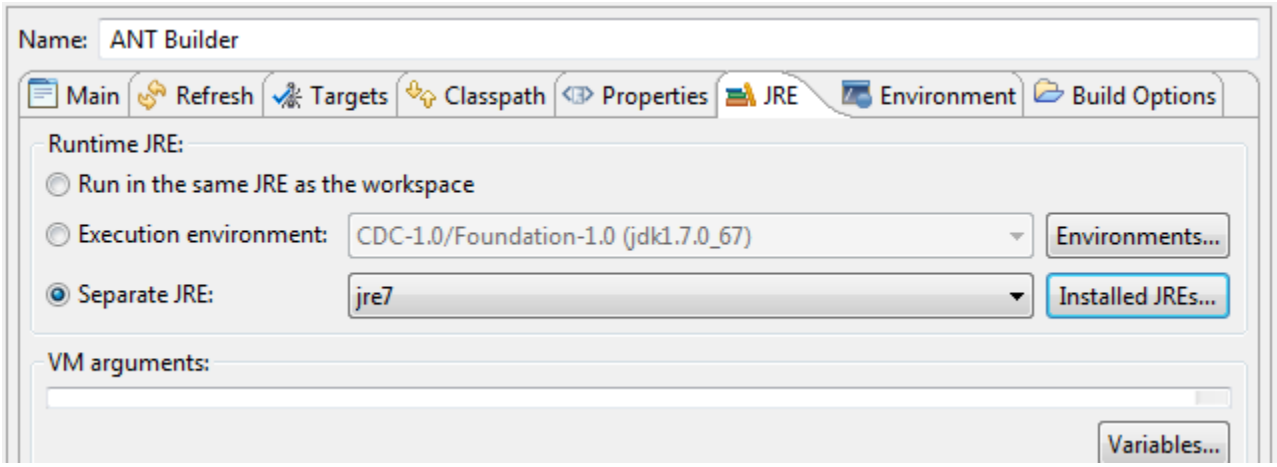
Click on **Finish** button.



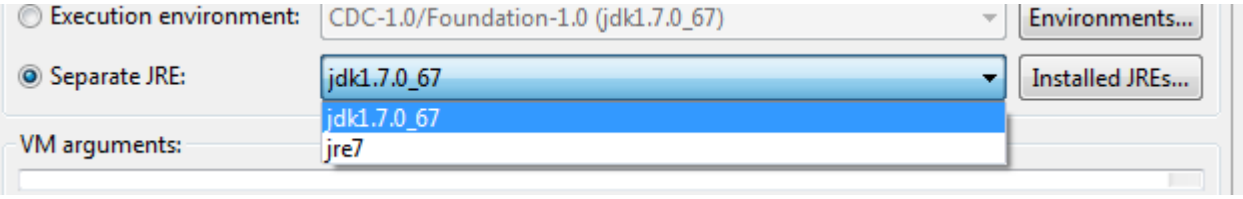
Now, make sure that the appropriate **JDK** is selected (in this case it would be *jdk1.7.0_67*).


Preferences (Filtered)
type filter text
Java
Installed JREs
Installed JREs
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
Type
jdk1.7.0_67 (...
C:\Program Files\Java\j...
Standard ...
jre7
C:\Program Files\Java\jre7
Standard VM
Add...
Edit...
Duplicate...
Remove
Search...
OK
Cancel

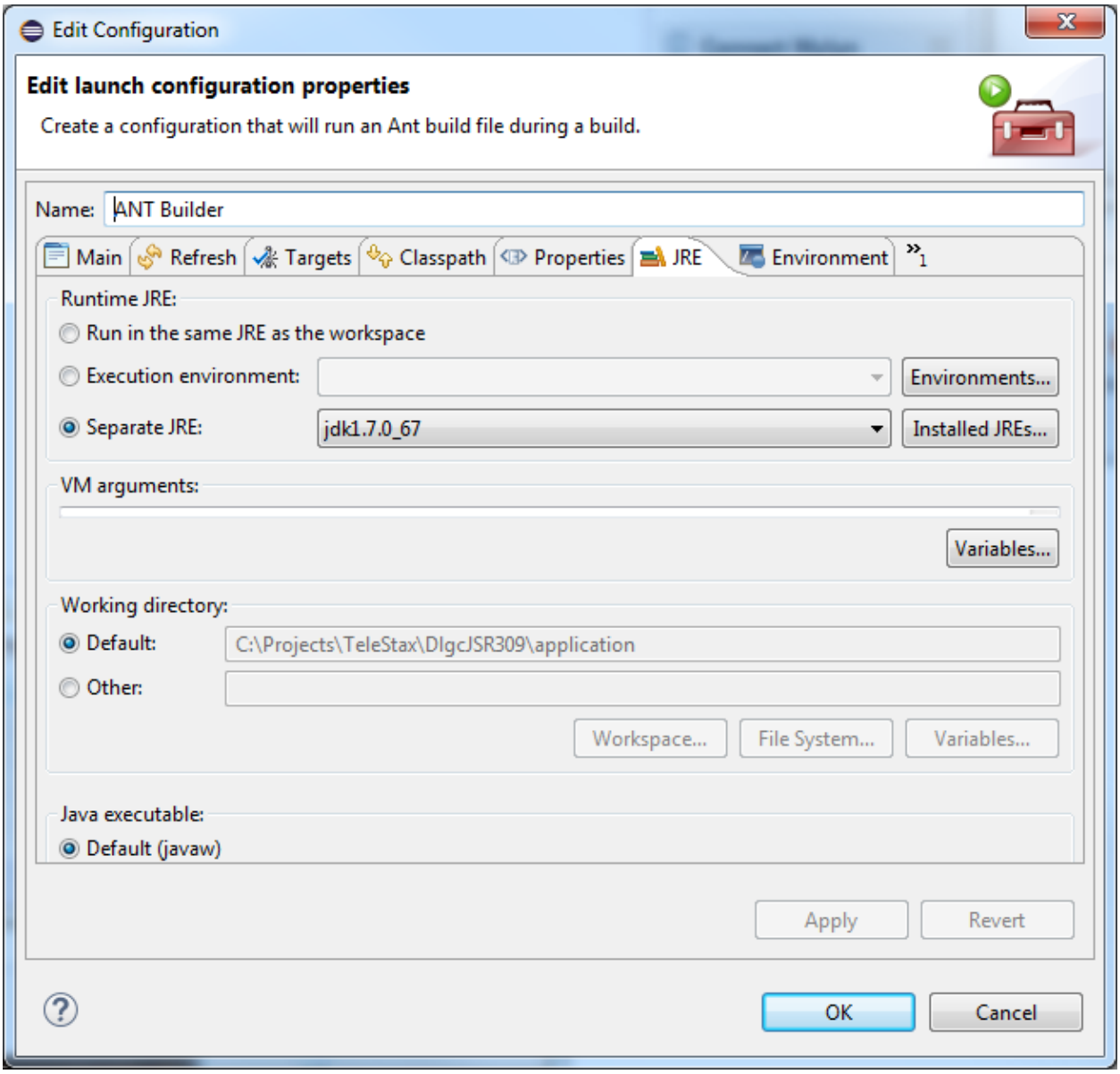
Click **OK**.


Name: ANT Builder
Main
Refresh
Targets
Classpath
Properties
JRE
Environment
Build Options
Runtime JRE:
Run in the same JRE as the workspace
Execution environment:
CDC-1.0/Foundation-1.0 (jdk1.7.0_67)
Environments...
Separate JRE:
jre7
Installed JREs...
VM arguments:
Variables...

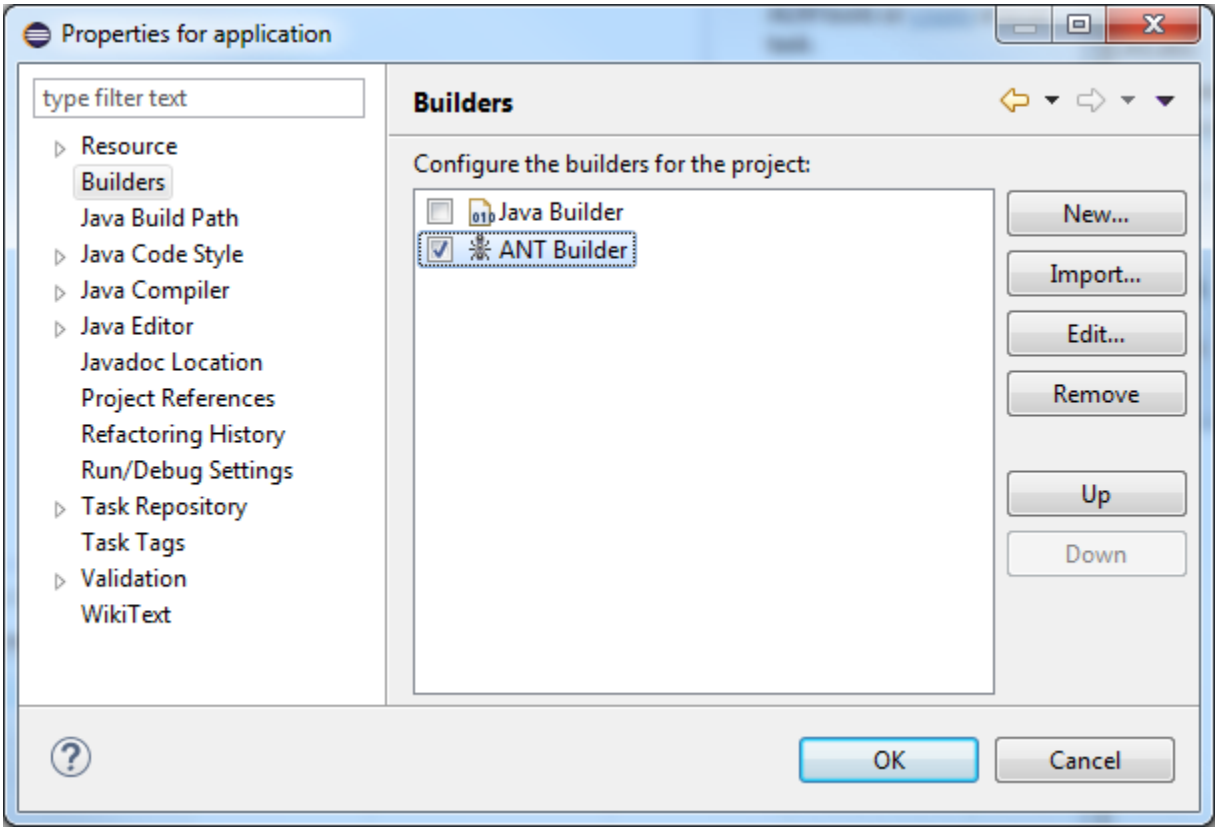
Now, make sure that the **Separate JRE** has the appropriate version of **JDK** selected:



Now, you have configured the appropriate **JDK** to be used by this project:

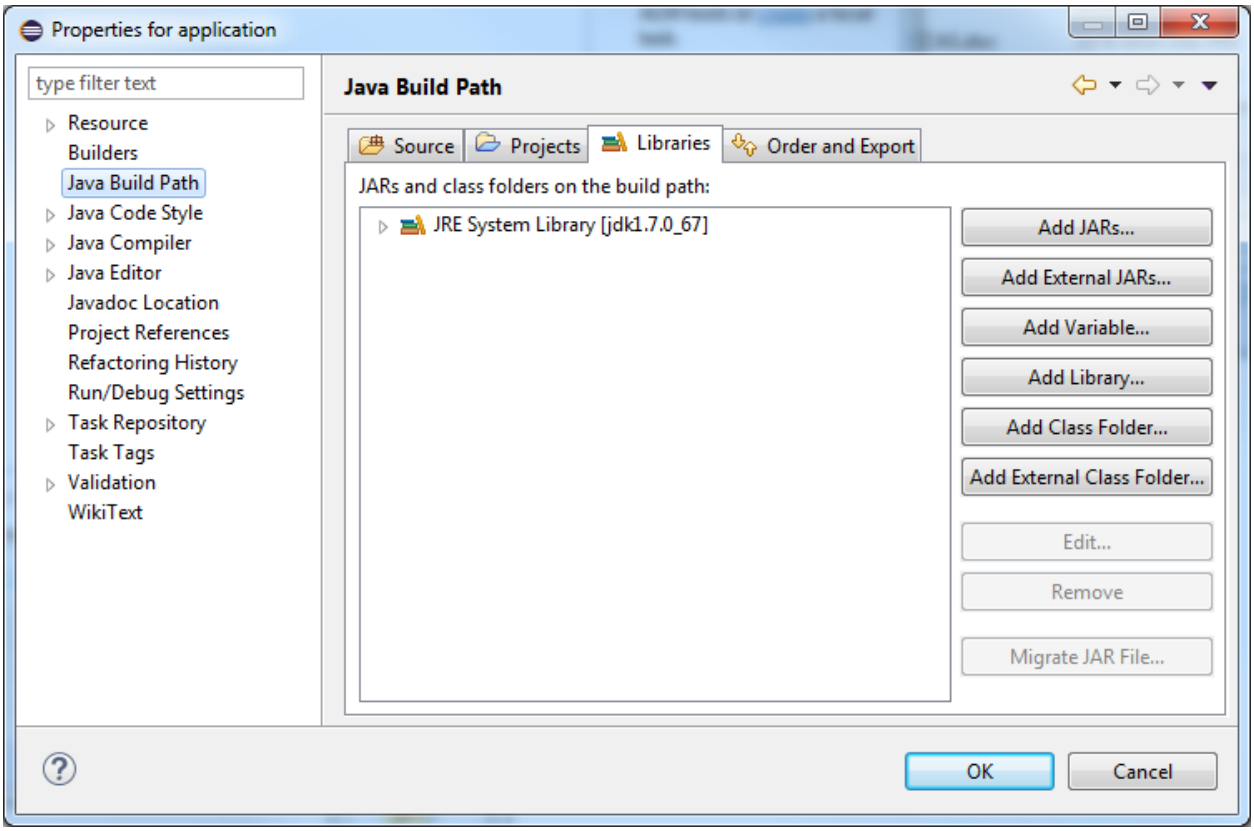


Click **Apply** and then click **OK**.



To ensure that the newly created builder (**ANT Builder**) is at the top of the list, click on **ANT Builder** and position it by clicking on **Up** button.

Next, the **Java Build Path** needs to be configured. Click on **Java Build Path** and then click on **Libraries** tab:



Click on **Add External JARs** button. Locate and click on "*DlgcJSR309/lib*" directory. Select all the files in that directory and click **Open**.

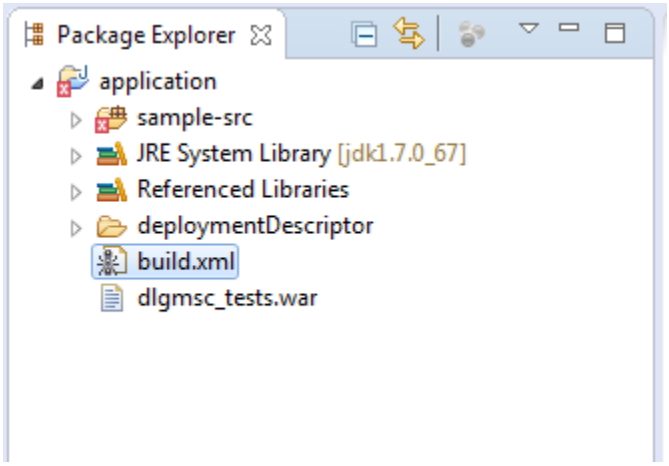
List of JAR files:

- *dlgcsmiltypes.jar*
- *geronimo-commonj_1.1_spec-1.0.jar*
- *jain-sip-sdp-1.2.91.jar*
- *json_simple-1.1.jar*
- *log4j-api-2.2.jar*
- *log4j-core-2.2.jar*
- *log4j-slf4j-impl-2.2.jar*
- *mscontrol.jar*
- *msmltypes.jar*
- *org.osgi-3.0.0.jar*
- *slf4j-api-1.7.5.jar*
- *xbean.jar*

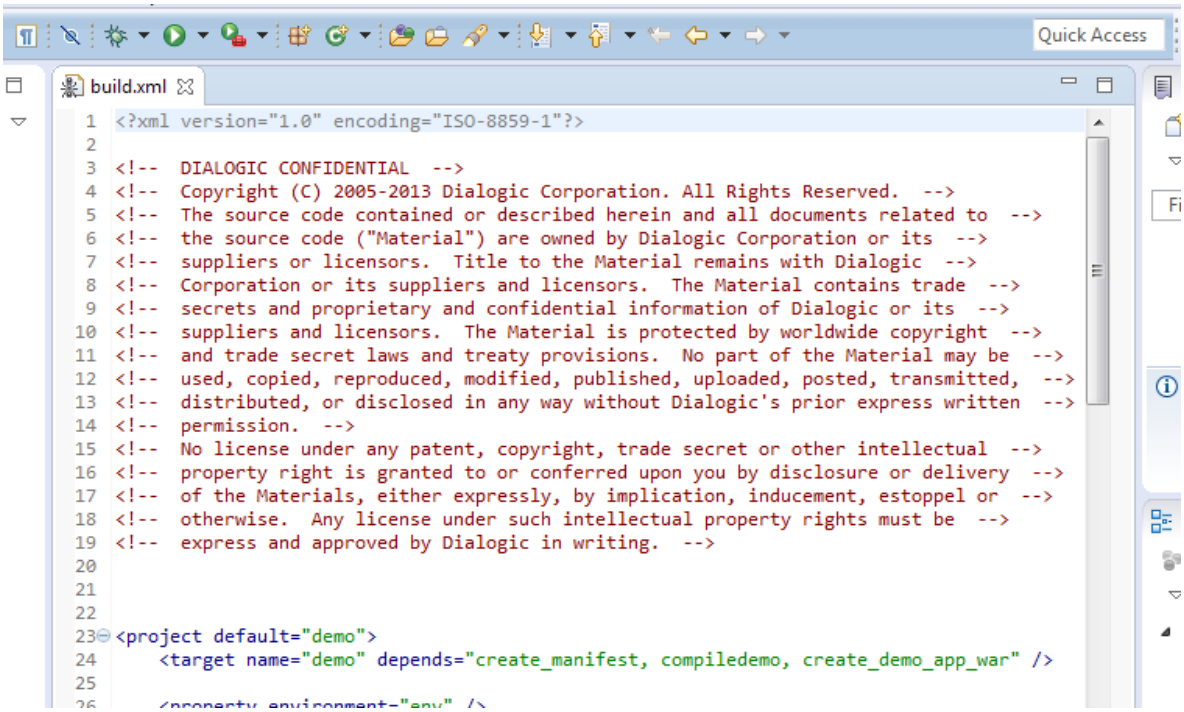
Also, you will need application server platform specific JAR files as referenced in the step below.

Now, click **OK**.

The last step is to modify *build.xml* project file to use appropriate Application Server platform specific libraries (JSR files). To do that, locate *build.xml* under newly configured project:



Double click on *build.xml* file in order to open it for editing:



Now, find the section as illustrated below and uncomment the appropriate platform section. In our example, TeleStax Mobicents Apache-Tomcat is used:

Before:

```
47                  <pathelement path="${DEPLOY_LIBS}/mscontrol.jar" />
48
49  <!-- Platform Specific Libraries to use - only use one set: -->
50  <!-- TeleStax-Mobicents-ApacheTomcat:
51                  <pathelement path="${DEPLOY_LIBS}/sip-servlets-spec-3.0.536.jar" />
52                  <pathelement path="${DEPLOY_LIBS}/servlet-api.jar" />
53  -->
54  <!-- TeleStax-TelScale-ApacheTomcat:
55                  <pathelement path="${DEPLOY_LIBS}/sip-servlets-spec-7.0.2.GA-TelScale.jar" />
56                  <pathelement path="${DEPLOY_LIBS}/servlet-api.jar" />
57  -->
58              </classpath>
59          </javac>
60      </target>
```

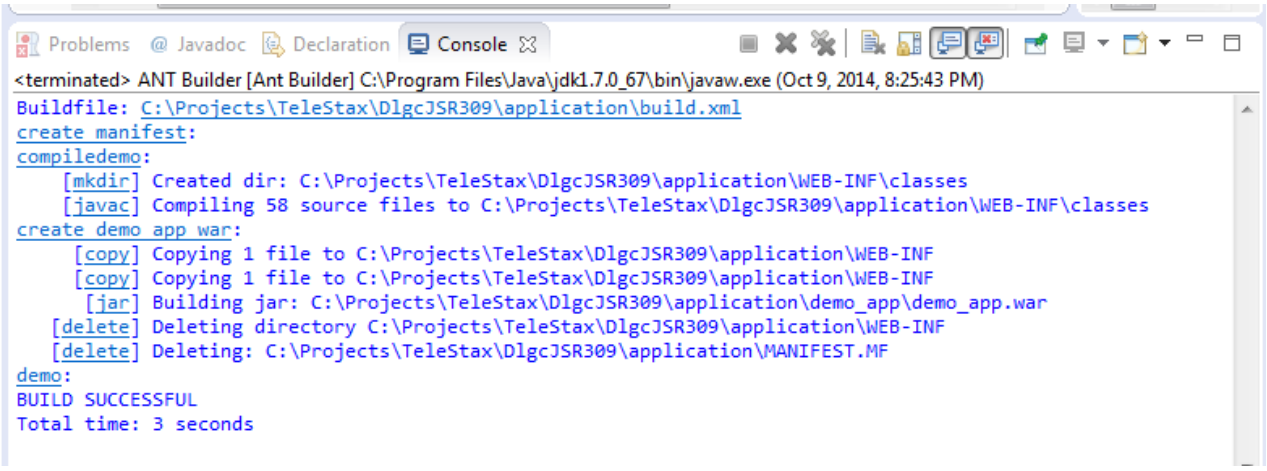
After:

```
47                  <pathelement path="${DEPLOY_LIBS}/mscontrol.jar" />
48
49  <!-- Platform Specific Libraries to use - only use one set: -->
50  <!-- TeleStax-Mobicents-ApacheTomcat:
51  -->             <pathelement path="${DEPLOY_LIBS}/sip-servlets-spec-3.0.536.jar" />
52                  <pathelement path="${DEPLOY_LIBS}/servlet-api.jar" />
53
54  <!-- TeleStax-TelScale-ApacheTomcat:
55                  <pathelement path="${DEPLOY_LIBS}/sip-servlets-spec-7.0.2.GA-TelScale.jar" />
56                  <pathelement path="${DEPLOY_LIBS}/servlet-api.jar" />
57  -->
58              </classpath>
59          </javac>
60      </target>
```

Now, click **File > Save**. The project configuration has now concluded.

Building the Project

After a successful project installation and configuration, a project can be built. In Eclipse, select the newly created project, then go under the **Project** menu and click on **Build All**. Successful build content will be shown in the **Console** view in Eclipse as follows:



The newly built application WAR file will be located under the "*DlgcJSR309\application\demo_app*" directory named *demo_app.war*. In order to deploy this application, follow the same deployment instructions as described in Installation and Configuration of JSR 309 Connector Demo.

Configuring Eclipse Project and TeleStax Application Server Deployed Application for Remote Debugging

In order to connect the newly created project to the deployed WAR file in Application Server for debugging purposes, developers need to follow two simple steps:

- Have Eclipse successfully build the JSR 309 Connector Demo Application WAR file and deploy it in the desired Application Server platform. Refer to Deploy JSR 309 Connector Demo Application.
- Configure Application Server platform for remote debugging.

Configuring Application Server Platform for Remote Debugging

- Stop Application Server.
- Edit examples:

```
(TelScale) /opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/bin/catalina.sh
(Mobicents) /opt/mss-3.0.536-apache-tomcat-7.0.50/bin/catalina.sh
```

- Script the file and add the following line enabling the remote debugging as illustrated below in **bold**:

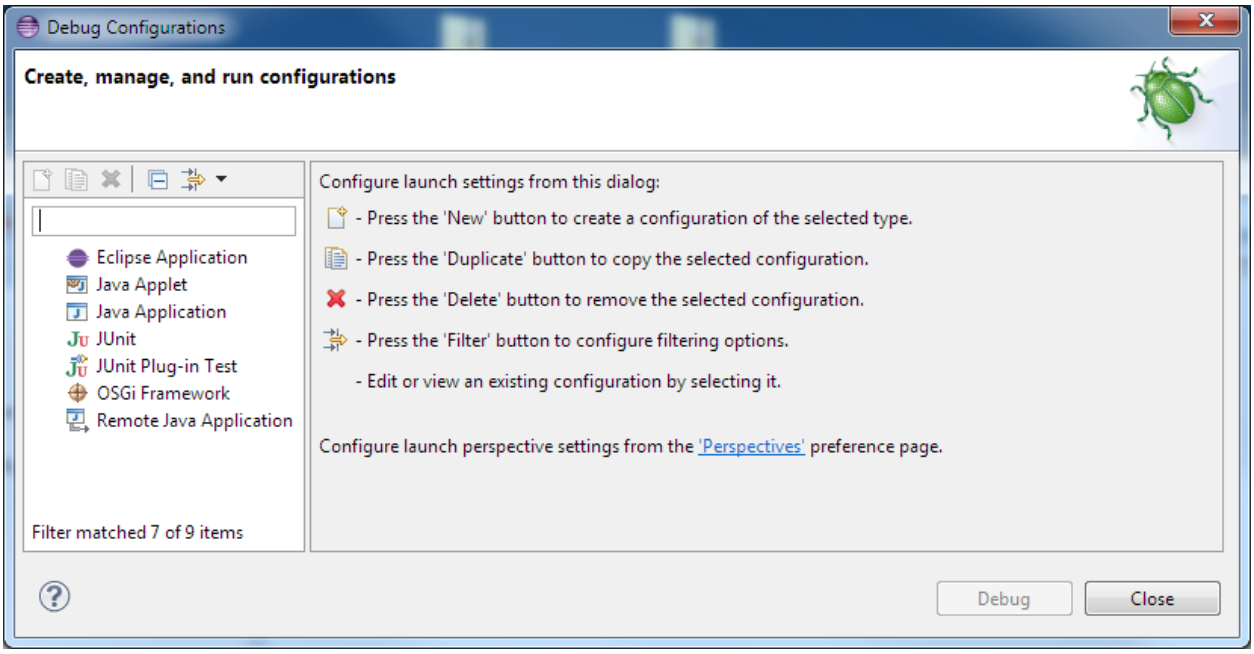
```
# Uncomment the following line to make the umask available when using the
# org.apache.catalina.security.SecurityListener
#JAVA_OPTS="$JAVA_OPTS -Dorg.apache.catalina.security.SecurityListener.UMASK=`umask`"
# ----- Execute The Requested Command -----------------------------------------

#Dialogic Additions
export APPSERVER_PLATFORM="TELESTAX"
export DIALOGIC_DEMO_PROPERTY_FILE=${CATALINA_HOME}/conf/Dialogic/dlgc_demos.properties
export DLG_PROPERTY_FILE=${CATALINA_HOME}/conf/Dialogic/dlgc_JSR309.properties
#Optional For Remote Debugging
CATALINA_OPTS="$CATALINA_OPTS -Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n"

# Bugzilla 37848: only output this if we have a TTY
if [ $have_tty -eq 1 ]; then
echo "Using CATALINA_BASE:   $CATALINA_BASE"
```

Note: The socket address specified above is 8000 but any port of choice can be used. Any port used needs to be enabled in a firewall in order to allow communication through it.

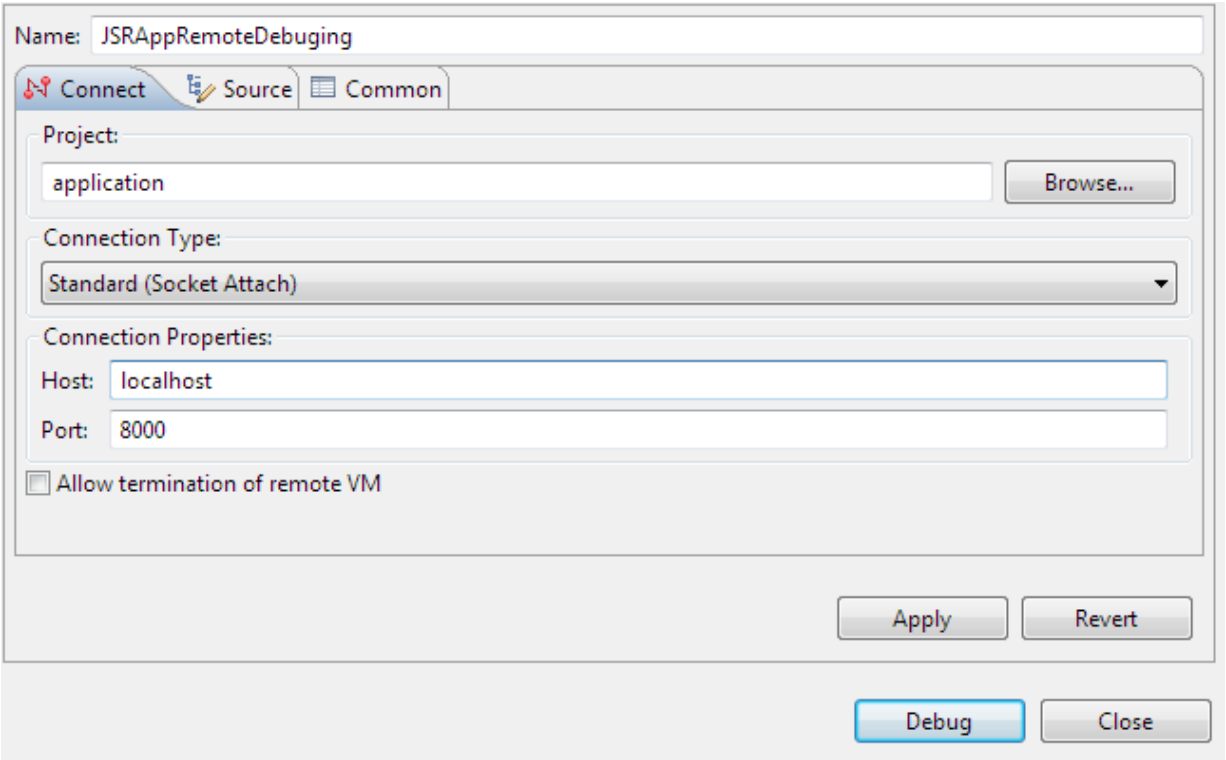
- Start Application Server and make sure there are no errors in the console.

Eclipse Project Configuration for Remote Debugging

To configure the existing and working JSR 309 Connector project, the remote debugging section needs to be configured. In Eclipse, go to the **Run** menu and click on **Debug Configurations**:



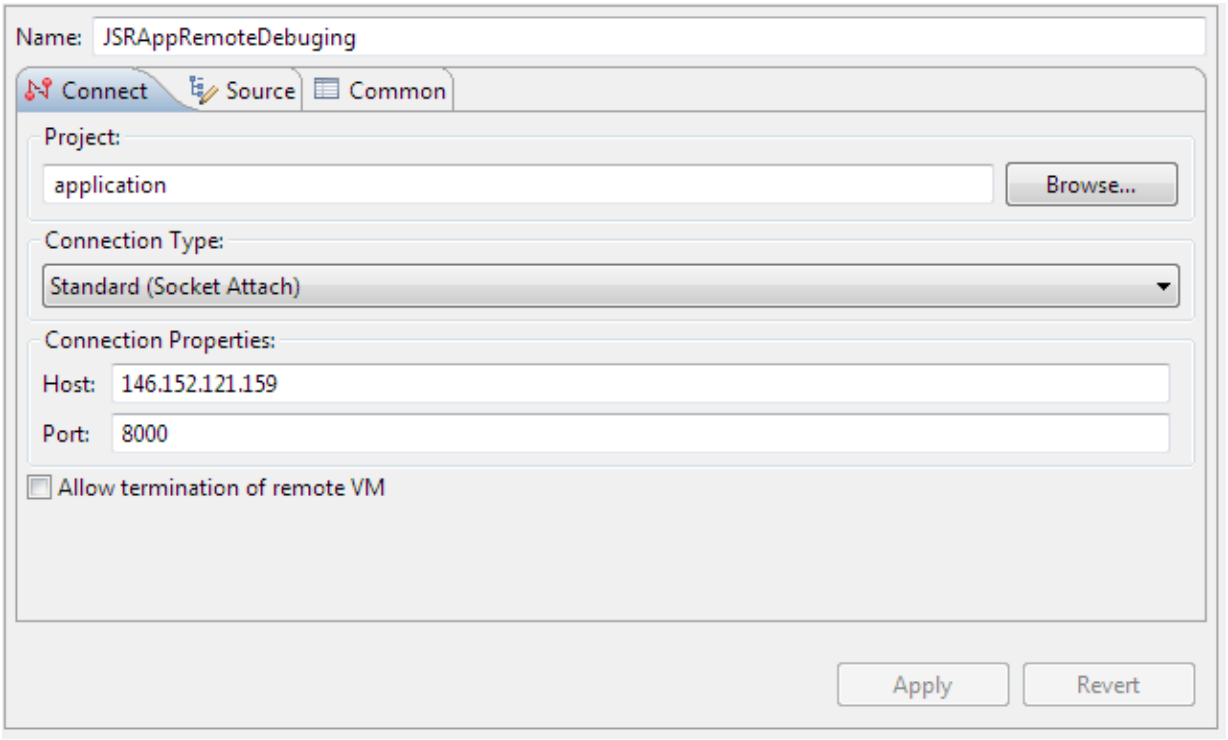
Double click on **Remote Java Application**.



Specify the **Name** for this remote debugging configuration (for example, JSRAppRemoteDebugging).

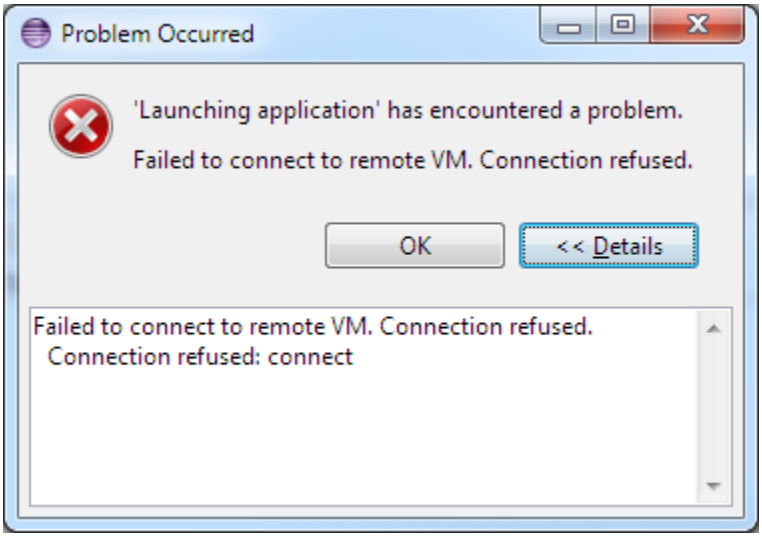
Under **Connection Properties**, specify the **Host** address, as well as the **Port** address of Application Server with deployed application for the debugger to connect to.

Below is an example of what your window would look like once information is added:



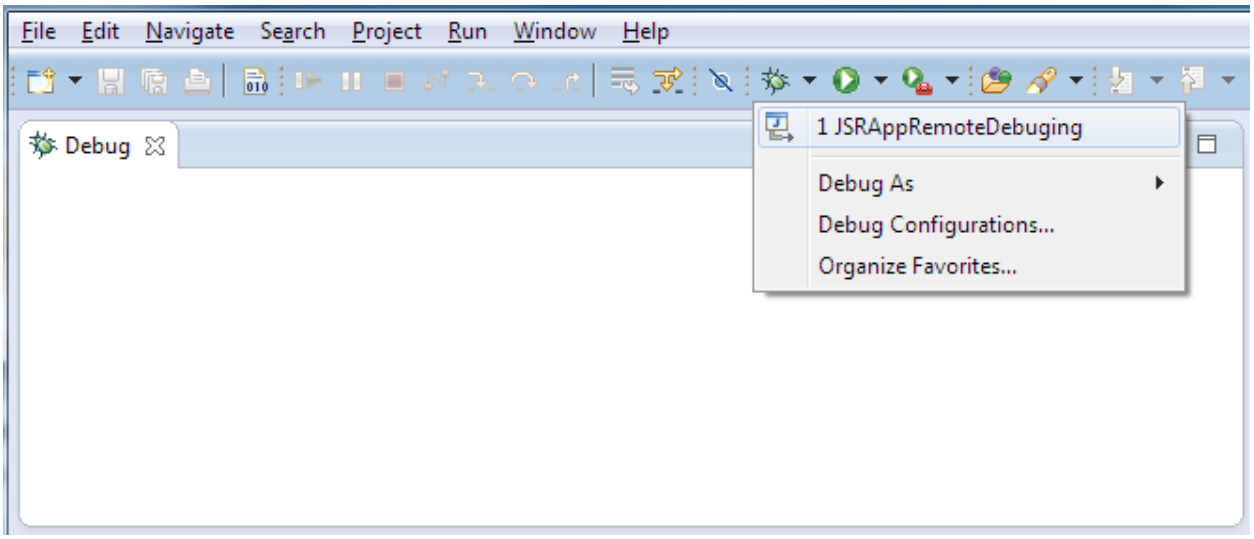
Once done, click **Apply** and then click **Debug**.

Application Server needs to be running at this point. If not, Eclipse will report a connection error message. If AS is running but Eclipse is still reporting a connection error, this could be due to either a port mismatch between Eclipse and AS firewall settings not allowing the specified port to be used; or there was simply a port conflict.

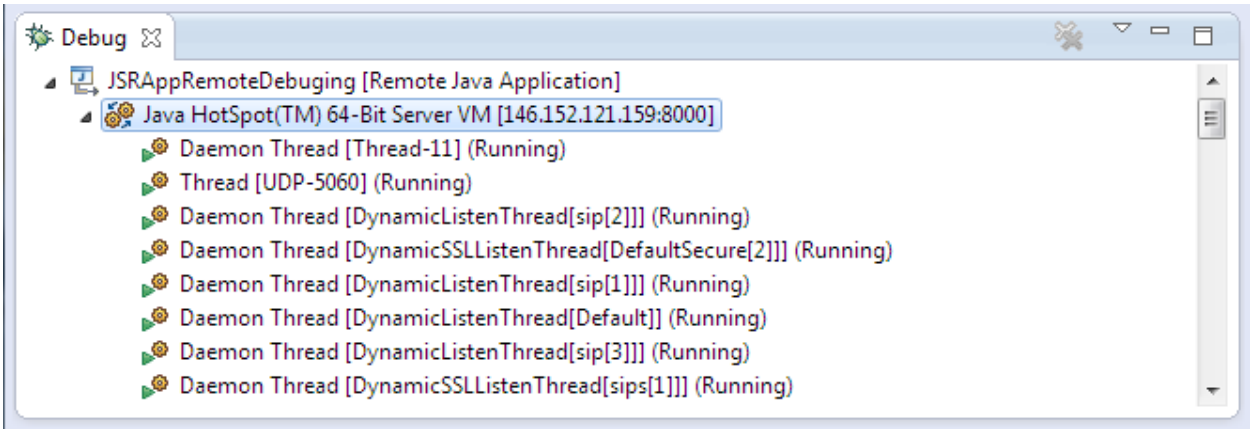


Now, open **debug perspective** in Eclipse (**Windows > Open Perspective > Debug**).

If nothing shows under the **Debug** section of a **debug perspective**, then a connection to AS has not been established. To connect/reconnect, go to the **debug icon** on the **toolbar** and choose the newly created **remote debugging configuration**:



Once the **remote debugging configuration** is selected and a connection is established, the content of the **Debug** window should show running threads:



Now, the Eclipse project is connected to the build application that is deployed in TeleStax Application Server.

7. Appendix A: JSR 309 Connector Environment Setup

This section describes, not going into platform details itself, a quick way to set up environment for JSR 309 Connector and JSR 309 application development.

For system requirements and supported platforms, see JSR 309 Connector Requirements.

This section does not go into details of Application Server platform, but simply will help build it quickly and be ready for use.

It should be noted that OS level configuration should include the following:

- Enable NTP (Network Time Protocol)
- Enable ports in firewall (if applicable)

  Note: The following IP ports will have to be enabled in the firewall for the system to operate correctly: 8080 (TCP), 9990 (TCP), 5080 (UDP & TCP), and optional remote debugging port 8000 (TCP).

If you need further details on TeleStax Application Server, visit: www.telestax.com.

Installing and Configuring the TeleStax Apache-Tomcat Application Server

Note: If you are familiar with TeleStax AS or are planning to deploy on an existing TeleStax setup, proceed to Installing the JSR 309 Connector.

Here are some highlights of the necessary steps:

- Pre-Installation Setup
- TeleStax Installation
- TeleStax Configuration
- Firewall Configuration
- TeleStax Startup
- TeleStax Verification

Pre-Installation Setup

Install OS supported by TeleStax – refer to www.telestax.com for details. For purpose of this documentation, CentOS 6.4/6.5 64-bit operating system with minimum installation options was used. Follow the steps below:

- Log into newly installed operating system and install zip/unzip package:

```
yum install zip unzip
```

- Copy and install latest 1.7 version of JDK rpm package which can be downloaded from www.oracle.com.

```
rpm –ivh jdk-7u60-linux-x64.rpm
```

- Under "*/root*" directory, edit *.bashrc* file and include the following export lines:

  (TelScale AS):

```
export JAVA_HOME=/usr/java/jdk1.7.0_67
export CATALINA_HOME=/opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50
```

(Mobicents AS):

```
export JAVA_HOME=/usr/java/jdk1.7.0_67
export CATALINA_HOME=/opt/mss-3.0.536-apache-tomcat-7.0.50
```

- Next, save the file and execute: *source .bashrc* for the changes to take an effect.
- Edit */etc/hosts* file and add a line at the very top of the file which corresponds to your systems IP address and the hostname specified during installation of the operating system and should match what is returned from "hostname" command when executed on the system command prompt in the example:

```
xxx.xxx.xxx.xxx TelScaleApacheTomcat
xxx.xxx.xxx.xxx MobicentsApacheTomcat
```

Note: This must be the first line in the */etc/hosts* file. If not, you might encounter a "503 Service Unavailable" error.

Run the following command at the prompt:

```
service network restart
```

TeleStax Installation

Copy the following to the system under the "*/opt*" directory and unzip them.

```
(TelScale AS) - TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50.zip
(Mobicents AS) - mss-3.0.536-apache-tomcat-7.0.50.zip
```

TeleStax Configuration

Edit the TeleStax specific configuration file by replacing any reference to "127.0.0.1" to system's hostname.

Note: The hostname can be retrieved by executing "hostname" at the command prompt.

Configuration file referenced below needs to be modified as shown below in **bold**:

```
/opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/conf/server.xml
/opt/mss-3.0.536-apache-tomcat-7.0.50/conf/server.xml
```

(before)

```
<!-- Define a SIP Connector on port 5080 -->
<Connector port="5080"
   ipAddress = "127.0.0.1"
   protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
   signalingTransport="udp"/>

   <!-- Define the default TCP SIP Connector -->
<Connector port="5080"
       ipAddress = "127.0.0.1"
       protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
       signalingTransport="tcp"/>

<!-- Define the default TLS SIP Connector -->
<Connector port="5081"
       ipAddress = "127.0.0.1"
       protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
       signalingTransport="tls"/>

<!-- Define the default SIP Over WebSockets Connector -->
<Connector port="5082"
       ipAddress = "127.0.0.1"
       protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
       signalingTransport="ws"/>
```

(after)

```
<!-- Define a SIP Connector on port 5080 -->
<Connector port="5080"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="udp"/>

<!-- Define the default TCP SIP Connector -->
<Connector port="5080"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="tcp"/>

<!-- Define the default TLS SIP Connector -->
<Connector port="5081"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="tls"/>

<!-- Define the default SIP Over WebSockets Connector -->
<Connector port="5082"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="ws"/>
```

Note: If the Application Server is configured in a cloud where it has an internal (private) IP address and externally bound IP address, the user will need to configure the following in order for AS to use external IP address instead of private one:

```
staticServerAddress="xxx.xxx.xxx.xxx"
staticServerPort="xxxx"
useStaticAddress="true"
```

Example:

```
<!-- Define a SIP Connector on port 5080 -->
<Connector port="5080"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="udp"
staticServerAddress="xxx.xxx.xxx.xxx"
staticServerPort="5080"
useStaticAddress="true"/>

<!-- Define the default TCP SIP Connector -->
<Connector port="5080"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="tcp"
staticServerAddress="xxx.xxx.xxx.xxx"
staticServerPort="5080"
useStaticAddress="true"/>

<!-- Define the default TLS SIP Connector -->
<Connector port="5081"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="tls"
staticServerAddress="xxx.xxx.xxx.xxx"
staticServerPort="5081"
useStaticAddress="true"/>

<!-- Define the default SIP Over WebSockets Connector -->
<Connector port="5082"
ipAddress = "TelScaleApacheTomcat"
protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
signalingTransport="ws"
staticServerAddress="xxx.xxx.xxx.xxx"
staticServerPort="5082"
useStaticAddress="true"/>
```

Firewall Configuration

At this point, we need to allow several ports to go through the firewall. Ports which are going to be in use are: 8080, 9990, 5080, and optional remote debugging port: 8000. To do this, simply edit */etc/sysconfig/iptables* files and add the lines in bold, making sure they are added before the REJECT lines:

```
vi /etc/sysconfig/iptables
```

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 9990 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5080 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 5080 -j ACCEPT
#optional port needs to be opened if remote debugging is required.
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8000 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Save the file and restart the firewall for the changes to take effect by executing the following command:

```
service iptables restart
```

TeleStax Startup

Now, you are ready to run Application Server. Go to:

```
/opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/bin/
/opt/mss-3.0.536-apache-tomcat-7.0.50/bin/
```

Then, execute the following command:

```
./catalina.sh run
```

```
date=201402261140)
2014-03-28 17:47:11,391 INFO  [Version] (main) TelScale Sip Servlets 7.0.1.GA-TelScale (build: Git Hash=r29c09ffda8e873a48a0208062f0cf64636e5b431 date=20140226
1140) Started.
2014-03-28 17:47:11,391 INFO  [Version] (main)
================================================================================
==                                                                            ==
==                      Thank you for running TelScale                        ==
==    Carrier Grade Communications Platform by the creators of Mobicents      ==
==                    Copyright 2011-2013 Telestax, Inc.                      ==
==                         http://www.telestax.com/                           ==
==                                                                            ==
================================================================================

2014-03-28 17:47:11,393 WARN  [SipStackImpl] (main) Could not register the stack as a Notification Listener of jboss.system:service=Logging,type=Log4jService r
untime changes to log4j.xml won't affect SIP Stack Logging
Mar 28, 2014 5:47:11 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 43471 ms
```

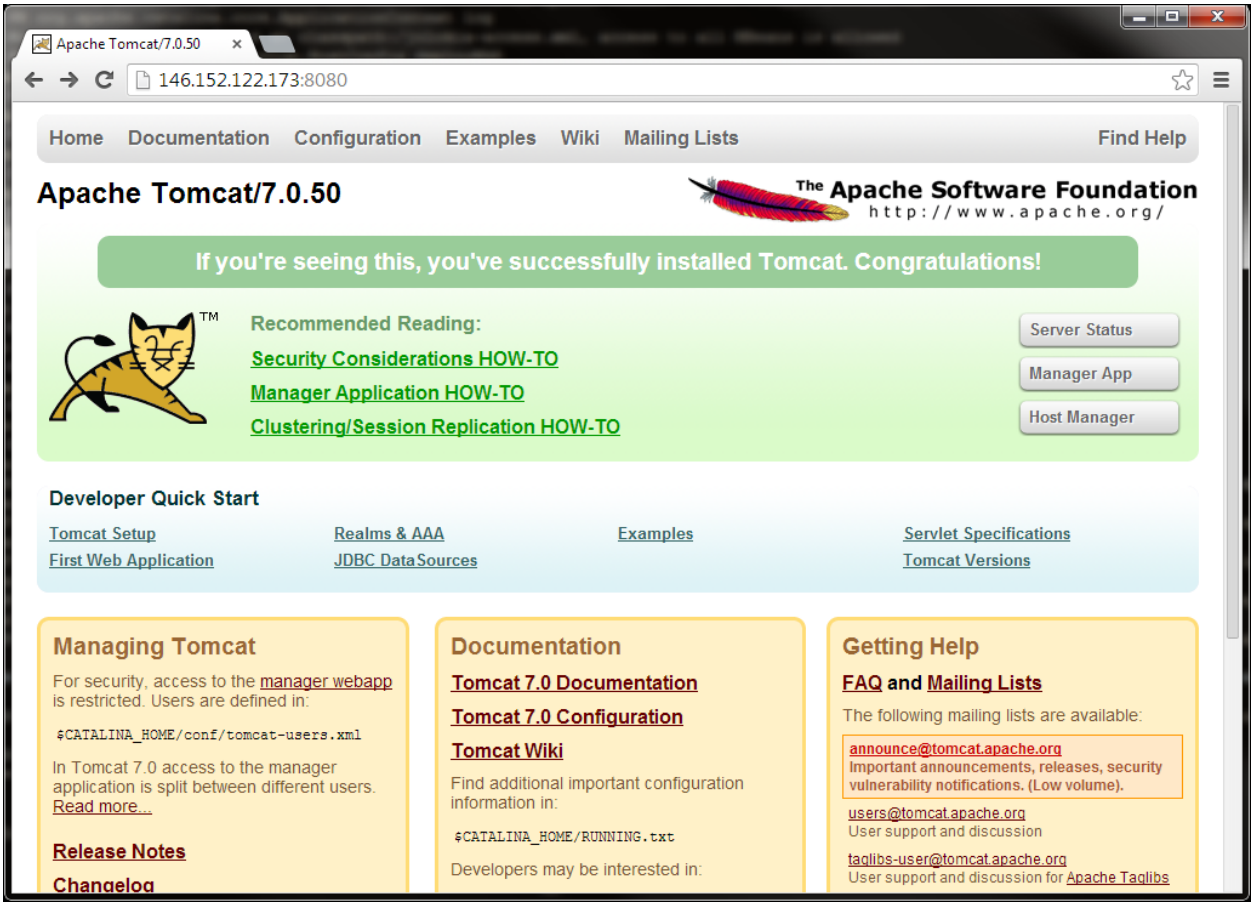
```
==========================================================================
==                                                                      ==
==              Thank you for running Mobicents Community code          ==
==   For Commercial Grade Support, please request a TelScale Subscription   ==
==                        http://www.telestax.com/                      ==
==                                                                      ==
==========================================================================

2014-04-22 11:59:44,936 WARN  [SipStackImpl] (main) Could not register the stack as a Notification L
istener of jboss.system:service=Logging,type=Log4jService runtime changes to log4j.xml won't affect
SIP Stack Logging
Apr 22, 2014 11:59:44 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 23281 ms
```

To stop the service, press **Ctrl-C**.

TeleStax Verification

Once application server service is started, the access to TeleStax Apache-Tomcat Web Administration can be done from any browser by going to the following URL:

http://<as_ip_address>:8080



With newly installed AS, there will be no access to any of the configuration sections. The following steps need to be taken to allow access:

- Edit *tomcat-users.xml* files to allow WebUI access:

```
/opt/TelScale-SIP-Servlets-7.0.2.GA-apache-tomcat-7.0.50/conf/tomcat-users.xml
/opt/mss-3.0.536-apache-tomcat-7.0.50/conf/tomcat-users.xml
```

- Enter the following line:

```
<!--
  NOTE:  The sample user and role entries below are wrapped in a comment
  and thus are ignored when reading this file. Do not forget to remove
  <!.. ..> that surrounds them.
-->
<role rolename="manager-gui"/>
<user username="admin" password="admin" roles="manager-gui"/>
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
```

Now, you be able to access any configuration screen by using the above credentials (admin/admin) as an example.

8. Appendix B: Redundant Media Servers Configuration

The Redundant Media Server feature provided by the JSR 309 Connector supports hot active/standby redundancy. The JSR 309 Connector allows for "n" number of PowerMedia XMS systems to be configured where only one is an active Media Server and the rest are considered standby. This section explains how to configure hot active/standby.

A primary (hot active) or a single (redundancy is not used) PowerMedia XMS is defined in the *dlgc_JSR309.properties* file:

```
#################### Dialogic PowerMedia XMS Media Server Configuration ########################
# Configuration of PowerMedia XMS Media Server
mediaserver.1.sip.address=xxxx.xxxx.xxxx.xxxx
mediaserver.1.sip.port=xxxx

# mediaserver.count defines the number of PowerMedia XMS Media Servers used
# by the JSR 309 Connector.
# Supported values:
# 1: Specifies single Media Server configuration (Redundancy not used)
# <2-n>: Specifies ALL Media Servers to be used by connector (Redundancy ON).
#            NOTE: Requires Redundancy Configuration section to be configured.
# Default - 1:
mediaserver.count=1
```

Redundancy configuration can be found under the "Dialogic PowerMedia XMS MS Redundancy Configuration" section in the *dlgc_JSR309.properties* file:

```
#################### Dialogic PowerMedia XMS MS Redundancy Configuration ######################
# mediaserver.redundancy - turns redundancy feature "on" or "off"
# Default - off
mediaserver.redundancy=off

# Configuration of secondary set of PowerMedia XMS Media Server(s):
# NOTE: Configuration of primary PowerMedia XMS Media Server is defined in
# JSR 309 Connector Configuration section above as mediaserver.1.
# 1) Replicate the two lines below for each PowerMedia XMS used as secondary Media Server
# 2) change mediaserver.x to the next appropriate index
# 3) configure appropriate IP and PORT for each
# NOTE: number of Media Servers defined below has to match mediaserver.count parameter.
mediaserver.x.sip.address=xxx.xxx.xxx.xxx
mediaserver.x.sip.port=xxxx

# mediaserver.redundancy.check.interval (in milliseconds) defines a time interval used by
# by JSR 309 Connector for sending a keep alive ping
# Default - 5000
mediaserver.redundancy.check.interval=5000

# mediaserver.redundancy.nonprimary.discover.clock.cycle defines a number of cycles to delay
# keep alive ping for every secondary Media Server(s)
# NOTE: cycle is used for secondary Media Servers and only used on initial discovery,
# i.e., startup of JSR 309 Connector.
# cycle * interval = seconds to wait before pinging secondary Media Server
# Default - 1
mediaserver.redundancy.nonprimary.discover.clock.cycle=1
######## END - Dialogic PowerMedia XMS MS Redundancy Configuration ##############################
```

In the "Dialogic PowerMedia XMS Media Server Configuration" section:

- mediaserver.1.sip.address and mediaserver.1.sip.port need to be configured for hot active Media Server.
- mediaserver.count needs to specify a total number of Media Servers to be used by a connector (one designated as active hot and others designated as active standbys).

  For example, if there are 4 Media Servers to be used where one of them is hot active and other 3 are considered hot standbys mediaserver.count needs to be set to 4 (1 hot active and 3 hot standbys).

In the "Dialogic PowerMedia XMS MS Redundancy Configuration" section:

- mediaserver.redundancy needs to be set to "on".
- mediaserver.x.sip.address/port set of parameters need to be configured for hot standby Media Servers where x is from 2-4 (if total of 4 Media Servers are used).

Optional:

- mediaserver.redundancy.check.interval parameter defines a time interval in milliseconds for the JSR 309 Connector to send a keep alive ping to all configured Media Servers.

  ```
  Default - 5000
  ```

- mediaserver.redundancy.nonprimary.discover.clock.cycle parameter defines a number of cycles to delay a keep alive ping for every hot standby Media Server.

  Note: The cycle is used for secondary Media Servers and only used on initial discovery (i.e., startup of the JSR 309 Connector).

  ```
  cycle * interval = seconds to wait before pinging hot standby Media Server
  # Default - 1
  ```

For details on how the JSR 309 Connector Media Server Redundancy works, refer to the Redundant Media Servers Guidelines section in the *Dialogic® PowerMedia™ JSR 309 Connector Software Developer's Guide*.

9. Appendix C: Updating the JSR 309 Connector

The JSR 309 Connector comes as a set of JAVA library files (JAR). In the TeleStax Application Server, the required application files are part of the application WAR file structure.

To update the WAR file with a new JSR 309 Connector set of JAR files, the files need to be placed inside the application WAR file under the "*lib*" directory.

The JSR 309 Connector is a set of the following JAR files:

- *dlgmsc.jar*
- *msmltypes.jar*
- *dlgcsmiltypes.jar*

The *MANIFEST.MF* file has been included in the JSR 309 Connector JAR files with version and build number. The versions of these files need to be exactly the same and should not be mixed and matched with older or newer versions of the JAR files. In addition, the *MANIFEST.MF* file describes the version of PowerMedia XMS that the JSR 309 Connector was tested on.

To do so, open the *<ApplicationName>.war* file using an archiver utility (for example, a third-party program like 7-Zip) and replace with the new JSR 309 Connector JAR files under the "*WEB_INF/lib*" directory as shown below: